L 8

```
MMM         MMM     000000000     NNN         NNN  TTTTTTTTTTTTTTT    000000000     RRRRRRRRRRRR
MMM         MMM     000000000     NNN         NNN  TTTTTTTTTTTTTTT    000000000     RRRRRRRRRRRR
MMM         MMM     000000000     NNN         NNN  TTTTTTTTTTTTTTT    000000000     RRRRRRRRRRRR
MMMMMM   MMMMMM  000         000  NNN         NNN        TTT       000         000  RRR         RRR
MMMMMM   MMMMMM  000         000  NNN         NNN        TTT       000         000  RRR         RRR
MMMMMM   MMMMMM  000         000  NNN         NNN        TTT       000         000  RRR         RRR
MMM   MMM   MMM  000         000  NNNNNN      NNN        TTT       000         000  RRR         RRR
MMM   MMM   MMM  000         000  NNNNNN      NNN        TTT       000         000  RRR         RRR
MMM   MMM   MMM  000         000  NNNNNN      NNN        TTT       000         000  RRR         RRR
MMM         MMM  000         000  NNN   NNN   NNN        TTT       000         000  RRRRRRRRRRRR
MMM         MMM  000         000  NNN   NNN   NNN        TTT       000         000  RRRRRRRRRRRR
MMM         MMM  000         000  NNN   NNN   NNN        TTT       000         000  RRRRRRRRRRRR
MMM         MMM  000         000  NNN      NNNNNN        TTT       000         000  RRR   RRR
MMM         MMM  000         000  NNN      NNNNNN        TTT       000         000  RRR   RRR
MMM         MMM  000         000  NNN      NNNNNN        TTT       000         000  RRR   RRR
MMM         MMM  000         000  NNN         NNN        TTT       000         000  RRR      RRR
MMM         MMM  000         000  NNN         NNN        TTT       000         000  RRR      RRR
MMM         MMM  000         000  NNN         NNN        TTT       000         000  RRR      RKR
MMM         MMM     000000000     NNN         NNN        TTT          000000000     RRR         RRR
MMM         MMM     000000000     NNN         NNN        TTT          000000000     RRR         RRR
MMM         MMM     000000000     NNN         NNN        TTT          000000000     RRR         RRR
```

```
**FILE**ID**MONITOR



          MM      MM    OOOOOO    NN      NN    IIIIII    TTTTTTTTTT    OOOOOO    RRRRRRRR
          MM      MM    OOOOOO    NN      NN    IIIIII    TTTTTTTTTT    OOOOOO    RRRRRRRR
          MMMM  MMMM  OO      OO  NN      NN      II          TT      OO      OO  RR      RR
          MMMM  MMMM  OO      OO  NN      NN      II          TT      OO      OO  RR      RR
          MM  MM  MM  OO      OO  NNNN    NN      II          TT      OO      OO  RR      RR
          MM  MM  MM  OO      OO  NNNN    NN      II          TT      OO      OO  RR      RR
          MM      MM  OO      OO  NN  NN  NN      II          TT      OO      OO  RRRRRRRR
          MM      MM  OO      OO  NN  NN  NN      II          TT      OO      OO  RRRRRRRR
          MM      MM  OO      OO  NN    NNNN      II          TT      OO      OO  RR  RR
          MM      MM  OO      OO  NN    NNNN      II          TT      OO      OO  RR  RR
          MM      MM  OO      OO  NN      NN      II          TT      OO      OO  RR    RR        ....
          MM      MM  OO      OO  NN      NN      II          TT      OO      OO  RR    RR        ....
          MM      MM    OOOOOO    NN      NN    IIIIII        TT        OOOOOO    RR      RR      ....
          MM      MM    OOOOOO    NN      NN    IIIIII        TT        OOOOOO    RR      RR      ....

          LL              IIIIII      SSSSSSSS
          LL              IIIIII      SSSSSSSS
          LL                II      SS
          LL                II      SS
          LL                II      SS
          LL                II      SS
          LL                II        SSSSSS
          LL                II        SSSSSS
          LL                II              SS
          LL                II              SS
          LL                II              SS
          LL                II              SS
          LLLLLLLLLL    IIIIII    SSSSSSSS
          LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000          .TITLE  MONITOR - VAX/VMS Performance Monitor Utility
0000          .IDENT  'V04-000'
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000
0000  ;++
0000  ; FACILITY:  VAX/VMS MONITOR Utility
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       This module is a collection of routines which are called by
0000  ;       the MONMAIN, REQUEST, and COLLEVT PL/I routines to do
0000  ;       various tasks, including data collection, terminal I/O
0000  ;       (through the use of the Screen Package) and maintenance
0000  ;       of statistics for screen displays.
0000  ;
0000  ; ENVIRONMENT:
0000  ;
0000  ;       Unprivileged user mode,
0000  ;       except for certain collection routines which
0000  ;       run in EXEC or KERNEL mode to access system
0000  ;       data bases.
0000  ;
0000  ; AUTHOR: Thomas L. Cafarella, April, 1981
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V03-027 TLC1090         Thomas L. Cafarella     02-Aug-1984     15:00
0000  ;               Correct ACCVIOs in SYSTEM and PROCESSES classes.
0000  ;
0000  ;       V03-026 TLC1087         Thomas L. Cafarella     25-Jul-1984     15:00
0000  ;               Default to /ALL when summarizing.
0000  ;
0000  ;       V03-025 TLC1086         Thomas L. Cafarella     24-Jul-1984     14:00
0000  ;               Make top summary work for SYSTEM class.
```

```
0000   ;
0000   ;       V03-024 TLC1085         Thomas L. Cafarella     22-Jul-1984     14:00
0000   ;               Calculate scale values for Free and Modified List bar graphs.
0000   ;
0000   ;       V03-024 TLC1083         Thomas L. Cafarella     20-Jul-1984     11:00
0000   ;               If counter value decreases, use 0 for delta.
0000   ;
0000   ;       V03-023 TLC1081         Thomas L. Cafarella     18-Jul-1984     11:00
0000   ;               Correct use of R6 in PROCESSES /TOP display.
0000   ;
0000   ;       V03-022 TLC1078         Thomas L. Cafarella     11-Jul-1984     11:00
0000   ;               Use SCSNODE node name if present before using SYS$NODE.
0000   ;
0000   ;       V03-021 TLC1072         Thomas L. Cafarella     17-Apr-1984     11:00
0000   ;               Add volume name to DISK display.
0000   ;
0000   ;       V03-020 PRS1019         Paul R. Senn            11-Apr-1984     16:00
0000   ;               Fix /SUMMARY for SYSTEM class.
0000   ;
0000   ;       V03-020 PRS1018         Paul R. Senn            11-Apr-1984     9:00
0000   ;               Display CPU busy instead of CPU idle in SYSTEM class.
0000   ;
0000   ;       V03-020 TLC1066         Thomas L. Cafarella     01-Apr-1984     11:00
0000   ;               Add SYSTEM class.
0000   ;
0000   ;       V03-020 TLC1063         Thomas L. Cafarella     3-Apr-1984      13:00
0000   ;               Add check to ensure that a counter which is re-inited to
0000   ;               zero will not cause an **** to be displayed.
0000   ;
0000   ;       V03-020 PRS1015         Paul R. Senn            3-Apr-1984      15:00
0000   ;               add shared error message capability
0000   ;
0000   ;       V03-020 TLC1062         Thomas L. Cafarella     31-Mar-1984     23:00
0000   ;               Fix bug causing summary averages to be displayed as zeroes
0000   ;               for homogeneous classes.
0000   ;
0000   ;       V03-019 TLC1061         Thomas L. Cafarella     18-Mar-1984     11:00
0000   ;               Identify dual-path 'isks by allocation class.
0000   ;
0000   ;       V03-019 TLC1060         Thomas L. Cafarella     12-Mar-1984     11:00
0000   ;               Make multi-file summary work for homogeneous classes.
0000   ;
0000   ;       V03-018 PRS1008         Paul R. Senn            17-FEB-1984     14:00
0000   ;               Move GET_BUFFERS and associated subroutines into separate
0000   ;               module.
0000   ;
0000   ;       V03-018 PRS1006         Paul R. Senn            17-FEB-1984     14:00
0000   ;               Add support for "computed" items
0000   ;
0000   ;       V03-018 TLC1052         Thomas L. Cafarella     17-Feb-1984     11:00
0000   ;               Add multi-file summary capability.
0000   ;
0000   ;       V03-017 PRS1005         Paul R. Senn            13-JAN-1984     10:00
0000   ;               Allow flexible spacing between screen items
0000   ;
0000   ;       V03-016 TLC1051         Thomas L. Cafarella     11-Jan-1984     11:00
0000   ;               Add consecutive number to class header record.
```

```
0000  ;
0000  ;       V03-016 PRS1002         Paul R. Senn            29-Dec-1983     16:00
0000  ;               Add YES and NO global equated symbols, which are no longer
0000  ;               defined globally by MONMAIN.
0000  ;
0000  ;       V03-016 PRS1000         Paul R. Senn            15-Dec-1983     16:00
0000  ;               For cases where one display event may involve multiple
0000  ;               screens of data (such as PROCESSES and Homogeneous
0000  ;               classes), make the wait between screens = VIEWING_TIME,
0000  ;               instead of a constant of 2 seconds.
0000  ;
0000  ;       V03-015 TLC1050         Thomas L. Cafarella     06-Dec-1983     11:00
0000  ;               Change directory information in DLOCK class.
0000  ;
0000  ;       V03-015 TLC1049         Thomas L. Cafarella     10-Oct-1983     15:00
0000  ;               Position TOP bar display properly.
0000  ;
0000  ;       V03-014 TLC1048         Thomas L. Cafarella     11-Sep-1983     12:00
0000  ;               Remove UIC from PROCESSES displays.
0000  ;
0000  ;       V03-013 TLC1047         Thomas L. Cafarella     09-Sep-1983     10:00
0000  ;               De-establish CTRL/W handler to get back AST quota.
0000  ;
0000  ;       V03-012 TLC1043         Thomas L. Cafarella     30-Jul-1983     13:00
0000  ;               Eliminate special characters from node name.
0000  ;
0000  ;       V03-012 SPC0004         Stephen P. Carney       24-Jun-1983     16:00
0000  ;               Add SCS display subroutine for new SCS Class.  Add FAO and
0000  ;               ASCII string for SCS class.
0000  ;
0000  ;       V03-011 TLC1042         Thomas L. Cafarella     19-Jun-1983     15:00
0000  ;               Add /ITEM qualifier for homogeneous classes.
0000  ;
0000  ;       V03-011 TLC1039         Thomas L. Cafarella     15-Jun-1983     15:00
0000  ;               Add DECnet node name to heading.
0000  ;
0000  ;       V03-011 TLC1036         Thomas L. Cafarella     10-Jun-1983     15:00
0000  ;               Properly recognize Revision Level 0.
0000  ;
0000  ;       V03-010 TLC1035         Thomas L. Cafarella     06-Jun-1983     15:00
0000  ;               Add homogeneous class type and DISK class.
0000  ;
0000  ;       V03-009 TLC1030         Thomas L. Cafarella     25-Apr-1983     10:00
0000  ;               Initialize MIN and MAX buffers.
0000  ;
0000  ;       V03-009 TLC1029         Thomas L. Cafarella     21-Apr-1983     10:00
0000  ;               Correctly calculate "Interrupt Stack" string.
0000  ;
0000  ;       V03-008 TLC1028         Thomas L. Cafarella     14-Apr-1983     16:00
0000  ;               Add interactive user interface.
0000  ;
0000  ;       V03-008 TLC1027         Thomas L. Cafarella     14-Apr-1983     16:00
0000  ;               Enhance file compatibility features.
0000  ;
0000  ;       V03-008 SPC0001         Stephen P. Carney       25-Mar-1983     15:00
0000  ;               Add RWxxx and MUTEX states in place of MWAIT state.
0000  ;
```

```
0000  ;      V03-007 TLC1024          Thomas L. Cafarella     1-Mar-1983      11:00
0000  ;              Convert an IPID to an EPID before placing it on the FAO
0000  ;              stack (for MONITOR PROCESSES display).
0000  ;
0000  ;      V03-006 TLC1021          Thomas L. Cafarella     07-Jul-1982     16:00
0000  ;              Change $SSDEF symbols to GLOBAL since they are no longer
0000  ;              accessible at link time.
0000  ;
0000  ;      V03-005 TLC1018          Thomas L. Cafarella     12-Apr-1982     16:00
0000  ;              Collect all data at KERNEL mode instead of EXEC.
0000  ;
0000  ;      V03-004 TLC1016          Thomas L. Cafarella     02-Apr-1982     16:00
0000  ;              Replace references to EXE$GQ_SYSTIME with $GETTIM calls.
0000  ;
0000  ;      V03-004 TLC1015          Thomas L. Cafarella     01-Apr-1982     16:00
0000  ;              Change .PSECT options in order to group image sections.
0000  ;
0000  ;      V03-004 TLC1014          Thomas L. Cafarella     01-Apr-1982     13:00
0000  ;              Correct attached processor time reporting for MODES class.
0000  ;
0000  ;      V03-004 TLC1012          Thomas L. Cafarella     30-Mar-1982     13:00
0000  ;              Display user's comment string on screen line 5.
0000  ;
0000  ;      V03-003 TLC1008          Thomas L. Cafarella     28-Mar-1981     21:00
0000  ;              Fix to display first and last PROCESSES records on playback.
0000  ;
0000  ;      V03-002 TLC1002          Thomas L. Cafarella     20-Mar-1981     13:00
0000  ;              Change PROCESSES display from scroll-style to page-style to
0000  ;                      make it terminal-independent.
0000  ;
0000  ;              Widen working set field of PROCESSES display.
0000  ;
0000  ;              Reset DEC_CRT advanced video options at exit.
0000  ;
0000  ;      V03-001 TLC1001          Thomas L. Cafarella     16-Mar-1981     13:00
0000  ;              Add CTRL-W screen refresh support.
0000  ;
0000  ;--
```

```
    0000          .SBTTL  DECLARATIONS
00000000          .PSECT  MONDATA,QUAD,NOEXE
    0000  ;
    0000  ; INCLUDE FILES:
    0000  ;
    0000
    0000          $DCDEF                          ; define device class codes
    0000          $DIBDEF                         ; define device information block
    0000          $DSCDEF                         ; Descriptor Definitions
    0000          $IODEF                          ; insert I/O function codes
    0000          $IPLDEF                         ; define interrupt levels
    0000          $JPIDEF                         ; define GETJPI items
    0000          $PCBDEF                         ; process control block
    0000          $PHDDEF                         ; process header definitions
    0000          $PRDEF                          ; define processor registers
    0000          $PSLDEF                         ; define PSL fields
    0000          $RPBDEF                         ; define Restart Parameter Block
    0000          $RSNDEF                         ; define resource wait codes
    0000          $SCRDEF                         ; SCRPKG definitions
    0000          $SSDEF  GLOBAL                  ; define status codes
    0000          $STATEDEF                       ; define process state codes
    0000          $STSDEF                         ; define status fields
    0000          $SYIDEF                         ; define GETSYI item identifiers
    0000          $SHRDEF                         ; define shared error codes
    0000          $TTDEF                          ; define terinal functions
    0000
    0000          $CDBDEF                         ; Define Class Descriptor Block
    0000          $CDXDEF                         ; Define CDB Extension
    0000          $CHDDEF                         ; Define Change Descriptor
    0000          $IDBDEF                         ; define item descriptor block offsets
    0000          $MRBDEF                         ; Define Monitor Request Block
    0000          $MBPDEF                         ; Define Monitor Buffer Pointers
    0000          $MCADEF                         ; Define Monitor Communication Area
    0000          $MONDEF                         ; Monitor Recording File Definitions
    0000          $SCBDEF                         ; Define STATS Control Block
    0000          $MFSDEF                         ; Define Multi-File Summary Block
    0000          $TM4DEF                         ; Define temporary storage
    0000
    0000  ;
    0000  ; MACROS:
    0000  ;
    0000
    0000  ;
    0000  ; Local Macro Definitions
    0000  ;
    0000
    0000  ;
    0000  ; ALLOC Macro - Dynamically allocate space on the stack.
    0000  ;
    0000
    0000          .MACRO  ALLOC   LENGTH,RSLDESC,RSLBUF
    0000          SUBL    #<LENGTH+3>&<^C3>,SP
    0000          .IF     NB,RSLBUF
    0000          MOVL    SP,RSLBUF
    0000          .ENDC
    0000          PUSHL   SP
    0000          PUSHL   #LENGTH
```

```
          0000   268         MOVL    SP,RSLDESC
          0000   269         .ENDM   ALLOC
          0000   270
```

```
                  0000    272 ;
                  0000    273 ; EQUATED SYMBOLS:
                  0000    274 ;
                  0000    275
                  0000    276                                                     ; Codes for MOVE_CLASS_QUALS routine:
        00000001  0000    277 YES == 1                                            ; Yep
        00000000  0000    278 NO == 0                                             ; Nope
        00000000  0000    279 DEF_TO_CUR == 0                                     ; Move default values to current values
        00000001  0000    280 CUR_TO_ACT == 1                                     ; Move current values to active values
        00000002  0000    281 ACT_TO_CUR == 2                                     ; Move active values to current values
        00000003  0000    282 ALL_TO_ACT == 3                                     ; Move ALL statistic value to active
                  0000    283
        00000002  0000    284 COLL_BUFS == 2                                      ; Number of collection  buffers
        00000004  0000    285 REG_BUFS == 4                                       ; Number of "regular stats" buffers
        00000004  0000    286 PC_BUFS == 4                                        ; Number of "percent stats" buffers
                  0000    287
        7FFF7FFF  0000    288 LARGE_NO == ^X7FFF7FFF                              ; Very large number (integer or float)
                  0000    289
        0000000D  0000    290 CR = 13                                             ; carriage return
        0000001B  0000    291 ESC = 27                                            ; escape character
        0000000A  0000    292 LF = 10                                             ; line feed
        0000000F  0000    293 SI = 15                                             ; shift in (selects G0 char set on VT100)
        00000002  0000    294 BS_SECS = 2                                         ; seconds between screens (of a mult scr)
        00000028  0000    295 MAXBARS == 40                                       ; number of bar chars in horiz. graph
        0000001A  0000    296 MAXBARS_SYS == 26                                   ; same as above for SYSTEM display
        0000002A  0000    297 DEF_BAR = ^A/*/                                     ; default bar character (for hardcopy)
        00000061  0000    298 VID_BAR = ^A/a/                                     ; video terminal bar character
        000000C8  0000    299 MAX55HEIGHT = 200                                   ; max height of vertical bars (VT55)
        00000032  0000    300 STARTPOS = 50                                       ; stavt of bargraph position
        00000026  0000    301 START_XPOS = 34+4                                   ; starting x position for VT55 bar
        00000018  0000    302 VTHEIGHT == 24                                      ; number of rows on screen
        00000050  0000    303 VTWIDTH == 80                                       ; number of columns on screen
        00000008  0000    304 FIRST_DATA_LINE == 8                                ; line no of 1st data line on screen
        00000016  0000    305 LAST_DATA_LINE == 22                                ; line no of last data line on screen
        0000000F  0000    306 VTDATALINES == LAST_DATA_LINE - FIRST_DATA_LINE + 1
                  0000    307                                                     ; number of lines of actual data on screen
        0000004A  0000    308 VT55CWIDTH == 74                                    ; number of chars. on x axis
        000000EC  0000    309 VT55HEIGHT = 236                                    ; screen height in points
        00000200  0000    310 VT55WIDTH = 512                                     ; screen width in points
        00000005  0000    311 NAME_COL_TAB == 5                                   ; starting col no. for names -- tabular
        00000002  0000    312 NAME_COL_BAR == 2                                   ; starting col no. for names -- bar graph
        00000001  0000    313 NAME_COL_MFSUM == 1                                 ; starting col no. for names -- m.f. summ.
        0000004B  0000    314 PROC_LINE = 75                                      ; width of a PROCESSES display line
        00000000  0000    315 PUTS_REGSET = 0                                     ; value for reg graphics set to PUT_SCREEN
        00000000  0000    316 PUTS_ALTSET = 0                                     ; value for alt graphics set to PUT_SCREEN
        000000C8  0000    317 MAXELTS == 200                                      ; max no. of elements for a homog class
        00000190  0000    318 MAXELTS_MFS == MAXELTS*2                            ; max no. of homog elts in a m.f. summary
        0000001B  0000    319 MAX_ELIDLEN == 27                                   ; max length of an element ID for homog
        0000000F  0000    320 MAX_HOM_ITEMS == 15                                 ; max no. of items in a homog class
        00000200  0000    321 SCRDSC_SIZE = 512                                   ; screen package buffer size
        00001900  0000    322 FAOSTK_SIZE = MAXELTS*8*4                           ; size of FAOL parameter stack for data disp
        000006A4  0000    323 OUTDSC_SIZE = 1700                                  ; size of FAO output buffer for displays
        000005DC  0000    324 FAOCTR_SIZE == 1500                                 ; size in bytes of FAO control string
        0000001A  0000    325 PUTMSGSIZE = 26                                     ; size in longwords of $PUTMSG message arg v
        00007D00  0000    326 MAX_REC_SIZE == 32000                               ; maximum record size for PLAYBACK and RECOR
        00000000  0000    327 SYS_FAC_NO = 0                                      ; system facility code
        00000008  0000    328 TAB_LWORDS = 8                                      ; Define no. of FAOSTK longwords ...
```

```
                  0000                                                      ; ... for tabular display. Used by
                  0000                                                      ; ... FILL_HOMOG_SCREEN routine.
                  0000
        00000003  0000  BAR_LWORDS = 3                                      ' Define no. of FAOSTK longwords ...
                  0000                                                      ; ... for bar-graph display. Used by
                  0000                                                      ; ... FILL_HOMOG_SCREEN routine.
                  0000
                  0000  ;
                  0000  ; SYS$OUTPUT Device types for Monitor
                  0000  ; Loaded into SYSOUT_TYPE
                  0000  ;
        00000000  0000  DEC_CRT   = 0                                       ; includes VT100-compatible devices
        00000001  0000  VT5X      = 1                                       ; VT5x series (VT52 and VT55)
        00000002  0000  HARDCOPY  = 2                                       ; hardcopy terminal and disk file
        00000003  0000  OTHER_VID = 3                                       ; other video types
                  0000
                  0000  ;
                  0000  ; Monitor status codes which use shared error codes.
                  0000  ;
                  0000
                  0000  ;
                  0000  ;   **** NOTE ****      The MONITOR facility number is defined here and in
                  0000  ;                       MONMSG.MSG. Any change needs to be made in BOTH places.
                  0000  ;
                  0000
        000000CE  0000  FACNO = 206                                         ; local symbol for monitor facility #
                  0000
        00CE109A  0000  MNR$_OPENIN == <SHR$_OPENIN+STS$K_ERROR>!<FACNO@16>  ;open-input-file error
```

```
                                      0000  ;
                                      0000  ; OWN STORAGE:
                                      0000  ;
                                      0000
                            00000001  0000  NAME_COL::        .BLKB   1                  ; column number of name string
                            00000019  0001  BARSIZE::         .LONG   25                 ; width of bar graph in points (VT55)
                            00000006  0005  BARCHAR::         .BLKB   1                  ; bar graph character
                            0000000A  0006  BPU:              .BLKF   1                  ; no of bar chars per unit of output value
                            0000000E  000A  GMIN:             .BLKL   1                  ; min value which bar graph can represent
                            00000000  000E  CURGRAPH::        .LONG   0                  ; currently enabled VT55 graph
                            00000000  0012  CURXPOS:          .LONG   0                  ; current position for GRAPH6
                            00000000  0016  PROCS_PER_REC:    .LONG   0                  ; number of processes which can fit into a .
                                      001A                                               ; ... single PLAYBACK or RECORD file class r
                   00000000 00000000  001A  PROC_WRI_BUFD:    .LONG   0,0                ; PROCESSES write buffer descriptor
                            00000023  0022  TOP_PROCS:        .BLKB   1                  ; number of top processes to display
                            0000002B  0023  TOP_TIME:         .BLKQ   1                  ; collection time of most recent TOP
                                      002B                                               ; ... display (in system time units)
                            0000002F  002B  TOP_TICKS:        .BLKL   1                  ; number of clock ticks (10ms units)
                                      002F                                               ; ... covered by most recent TOP display
                                      002F
                            00000037  002F  S_TOP_TIME:       .BLKQ   1                  ; similar to above, but for SYSTEM
                                      0037                                               ; ... class
                            0000003B  0037  S_TOP_TICKS:      .BLKL   1                  ; ...
                                      003B                                               ; ...
                            0000007B  003B  SYS_TOP_VEC::     .BLKL   16                 ; vector of ptrs to SYSTEM TOP arrays
                            00000000  007B  SYS_DATA_ADDR::   .LONG   0                  ; addr of the SYSTEM TOP arrays
                            00000000  007F  SYS_DATA_LEN::    .LONG   0                  ; len of the SYSTEM TOP arrays
                                      0083
                            00000085  0083  SYS_BOX_STR_LEN:: .BLKW 1                    ; len of SYSTEM box string
                            00000089  0085  SYS_BOX_STR_ADDR:: .BLKL 1                   ; ... and its address
                            00000000  0089  NEWXPOS:          .LONG   0                  ; next point to plot for GRAPH6
                            00000091  008D  PTS_STAT:         .BLKL   1                  ; save area for SCRPKG return stati
                                      0091
                            00000000  0091  CTRLW_MASK:       .LONG   0                  ; mask required by QIO for out-of-band char
                            00800000  0095                    .LONG   ^X00800000         ; bit 23 representing CTRL-W
                                      0099
                            00000000  0099  CTRLZ_MASK:       .LONG   0                  ; mask required by QIO for out-of-band char
                            04000000  009D                    .LONG   ^X04000000         ; bit 26 representing CTRL-Z
                                      00A1
                            00000000  00A1  CTRLCZ_CHAN::     .LONG   0                  ; channel no. for CTRL-C and -Z
                            00000000  00A5  CTRLW_CHAN::      .LONG   0                  ; channel no. for CTRL-W
                                      00A9
                            FECED300  00A9  BET_SCREENS:      .LONG   -10*1000*1000*BS_SECS ; delta time between
                            FFFFFFFF  00AD                    .LONG   -1                 ; ... screens for single display event
                                      00B1
4F 43 24 53 59 53 000000B9'010E0000'  00B1  SYSCMD_DESC:      .ASCID  \SYS$COMMAND\      ; User command terminal
               44 4E 41 4D 4D         00BF
4F 4E 24 53 59 53 000000CC'010E0000'  00C4  SYSNOD_NAM:       .ASCID  \SYS$NODE\         ; DECnet node logical name
                           45 44      00D2
                                      00D4
                            00000000  00D4  VT55XINCR::       .LONG   0                  ; incr to next bar of graph
                                      00D8
                            000000E0  00D8  CB_ADDRS::        .BLKQ   1                  ; holds coll buffer addrs in GET_BUFFERS
                            000000E2  00E0  ITEM_TYPE:        .BLKW   1                  ; holds IDB item type code in FILL_DISP_BUFF
                            000000E4  00E2  HOMOG_TYPE:       .BLKW   1                  ; holds same as above in COMPUTE_STATS
                                      00E4
                            000000E8  00E4  PREV_PD:          .BLKL   1                  ; no. of processes displayed in previous int
```

C 13

```
                                00E8
                                00E8  PROMPT_STR::                                          ; string and descriptor for
                                00E8                                                        ; ... subcommand prompt string
                     0000000B'  00E8                      .LONG   20$-10$
                     000000F0'  00EC                      .LONG   10$
                         0A 0D  00F0  10$:                .BYTE   CR,LF
    20 3E 52 4F 54 49 4E 4F 4D  00F2                      .ASCII  \MONITOR> \
                                00FB  20$:
                                00FB
                                00FB  DYN_STRING::                                          ; dynamic string descriptor for
                                00FB                                                        ; ... use in MONMAIN.PLI
                          0000  00FB                      .WORD   0                         ; called routine will fill in length
                            0E  00FD                      .BYTE   DSC$K_DTYPE_T             ; string descriptor type
                            02  00FE                      .BYTE   DSC$K_CLASS_D             ; dynamic class
                      00000000  00FF                      .LONG   0                         ; called routine will fill in address
                                0103
```

```
                 0103   430 ;
                 0103   431 ; FAO-related buffers required for /DISPLAY
                 0103   432 ;
                 0103   433
      00001A03   0103   434 FAOSTK::         .BLKB   FAOSTK_SIZE      ; DISPLAY buffer containing data for input t
                 1A03   435
                 1A03   436 OUTDSC::                                 ; FAO output buffer descriptor for data disp
      000006A4   1A03   437                 .LONG   OUTDSC_SIZE
      00001A0B'  1A07   438                 .LONG   10$
      000020AF   1A0B   439 10$:            .BLKB   OUTDSC_SIZE
                 20AF   440
                 20AF   441 SCRDSC::                                 ; Screen Package buffer descriptor
      00000200   20AF   442                 .LONG   SCRDSC_SIZE
      000020B7'  20B3   443                 .LONG   10$
      000022B7   20B7   444 10$:            .BLKB   SCRDSC_SIZE
```

```
                     22B7     446 ;
                     22B7     447 ; Control and name strings for screen output.
                     22B7     448 ;
                     22B7     449 ; These strings contain embedded escape sequences. Before the strings
                     22B7     450 ; are sent to the Screen Package for output, the escape sequences are
                     22B7     451 ; interpreted (in the PUT_TO_SCREEN routine) and translated to general-
                     22B7     452 ; case screen package calls. The sequences embedded below are generally
                     22B7     453 ; equivalent to VT52 escape sequences, with the following exceptions:
                     22B7     454 ;
                     22B7     455 ;       1) Cursor addresses for both row and column are in the
                     22B7     456 ;          form acceptable to the Screen Package -- i.e., the top
                     22B7     457 ;          row is 1 and the left-hand column is 1.
                     22B7     458 ;
                     22B7     459 ;       2) ESC B means BOLD all text until an UNDO sequence.
                     22B7     460 ;
                     22B7     461 ;       3) ESC L means UNDERLINE all text until an UNDO sequence.
                     22B7     462 ;
                     22B7     463 ;       4) ESC R means REVERSE VIDEO all text until an UNDO sequence.
                     22B7     464 ;
                     22B7     465 ;       5) ESC U means UNDO all DEC_CRT advanced video attributes selected
                     22B7     466 ;                                    (i.e., BOLD, UNDERLINE, REVERSE, BLINK)
                     22B7     467 ;
                     22B7     468
                     22B7     469 CLRVT55::
                08'  22B7     470         .BYTE   10$-5$
                     22B8     471 5$:
              31 1B  22B8     472         .BYTE   ESC,^A/1/                    ; enter graphics mode
              20 41  22BA     473         .BYTE   ^A/A/,^X20                   ; turn off graphs
              30 49  22BC     474         .BYTE   ^A/I/,^X30                   ; turn off lines, cursors, etc.
              32 1B  22BE     475         .BYTE   ESC,^A/2/                    ; back to alphanumeric mode
                     22C0     476 10$:
                     22C0     477
                     22C0     478 NAMESTR::
                08'  22C0     479         .BYTE   10$-5$
                 0D  22C1     480 5$:     .BYTF   CR
           2A 23 21  22C2     481         .ASCII  '!#*'
                 0A  22C5     482         .BYTE   LF
           43 41 21  22C6     483         .ASCII  '!AC'
                     22C9     484 10$:
                     22C9     485
                     22C9     486 ;
                     22C9     487 ; Finish sequence -- set regular character set,
                     22C9     488 ; reset DEC_CRT (VT100) AVO characteristics, and
                     22C9     489 ; carriage return.
                     22C9     490 ;
                     22C9     491
                     22C9     492 FIN_SEQ::
                0C'  22C9     493         .BYTE   10$-5$
              47 1B  22CA     494 5$:     .BYTE   ESC,^A/G/       ; Set regular character set
  52 1B 4C 1B 42 1B  22CC     495         .BYTE   ESC,^A/B/,ESC,^A/L/,ESC,^A/R/
                     22D2     496                                 ; Set AVO char'cs so UNDO works
                 0D  22D2     497         .BYTE   CR              ; Need to send a character to set attribs
              55 1B  22D3     498         .BYTE   ESC,^A/U/       ; Undo DEC_CRT AVO characteristics
                 0D  22D5     499         .BYTE   CR
                     22D6     500 10$:
                     22D6     501
                     22D6     502 ;
                 22D6  ; Sequence to place the cursor on the bottom line and clear it.
                 22D6  ;
                 22D6
                 22D6  BOT_CURS::
             06' 22D6          .BYTE   10$-5$
     01 18 59 1B 22D7  5$:     .BYTE   ESC,^A/Y/,24,1          ; Position to bottom line on screen
           4B 1B 22DB          .BYTE   ESC,^A/K/               ; Clear to end of line
                 22DD  10$:
```

```
                                     22DD   ;
                                     22DD   ; Announcement string, including home and clear screen, set regular
                                     22DD   ; char set, and identification (top line of the screen).
                                     22DD   ;
                                     22DD
                                     22DD   ANNCE_STR::
                                 21' 22DD           .BYTE   10$-5$
      1D 01 59 1B 4A 1B 48 1B 47 1B  22DE   5$:     .BYTE   ESC,^A/G/,ESC,^A/H/,ESC,^A/J/,ESC,^A/Y/,1,29
69 6E 6F 4D 20 53 4D 56 2F 58 41 56  22E8           .ASCII  \VAX/VMS Monitor Utility\
   79 74 69 6C 69 74 55 20 72 6F 74  22F4
                                     22FF   10$:
                                     22FF
                                     22FF   ;
                                     22FF   ; Status string. This is bottom line (footing) of the screen.
                                     22FF   ; It contains indications for "PLAYBACK", "SUMMARIZING",
                                     22FF   ; and "RECORDING".
                                     22FF   ;
                                     22FF
                                     22FF   STATUS_STR::
                                 15' 22FF           .BYTE   10$-5$
                        01 18 59 1B  2300   5$:     .BYTE   ESC,^A/Y/,24,1
                           43 41 21  2304           .ASCII  \!AC\
                        22 18 59 1B  2307           .BYTE   ESC,^A/Y/,24,34
                           43 41 21  230B           .ASCII  \!AC\
                        46 18 59 1B  230E           .BYTE   ESC,^A/Y/,24,70
                           43 41 21  2312           .ASCII  \!AC\
                                     2315   10$:
                                     2315
                                     2315   ;
                                     2315   ; Title string, including cursor positioning and title (with
                                     2315   ; optional percent sign), centered and reversed.
                                     2315   ;
                                     2315
                                     2315   TITLF_STR::
                                 2B' 2315           .BYTE   10$-5$
                        01 02 59 1B  2316   5$:     .BYTE   ESC,^A/Y/,2,1           ; Position cursor
                        20 2A 23 21  231A           .ASCII  \!#* \                  ; Preceding blanks
                              52 1B  231E           .BYTE   ESC,^A/R/               ; Reverse-video
20 29 25 28 3C 23 21 20 43 41 21 20  2320           .ASCII  \ !AC !#<(%) !>\        ; Title text
                              3E 21  232C
                              55 1B  232E           .BYTE   ESC,^A/U/               ; Undo reverse-video
                        20 03 59 1B  2330           .BYTE   ESC,^A/Y/,3,32          ; Optional position cursor
41 21 20 65 64 6F 6E 20 6E 6F 20 20  2334           .ASCII  \  on node !AC\         ; Optional nodename text
                                 43  2340
                                     2341   10$:
                                     2341
                                     2341   ;
                                     2341   ; User's comment string, including cursor positioning
                                     2341   ; and comment string, centered and reversed.
                                     2341   ;
                                     2341
                                     2341   COMM_STR::
                                 11' 2341           .BYTE   10$-5$
                        01 05 59 1B  2342   5$:     .BYTE   ESC,^A/Y/,5,1           ; Position cursor
                        20 2A 23 21  2346           .ASCII  \!#* \                  ; Preceding blanks
                              52 1B  234A           .BYTE   ESC,^A/R/               ; Reverse-video
                     20 46 41 21 20  234C           .ASCII  \ !AF \                 ; Title text
```

```
                55 1B  2351    566          .BYTE   ESC,^A/U/                  ; Undo reverse-video
                       2353    567 10$:
```

```
                                     2353  ;
                                     2353  ; Date/time string, including cursor positioning.
                                     2353  ;
                                     2353
                                     2353  TIME_STR::
                                 0E' 2353             .BYTE   10$-5$
                         1F 04 59 1B  2354  5$:          .BYTE   ESC,^A/Y/,4,31
                            44 41 21  2358              .ASCII  \!AD\
                         2B 04 59 1B  235B              .BYTE   ESC,^A/Y/,4,43
                            44 41 21  235F              .ASCII  \!AD\
                                     2362  10$:
                                     2362
                                     2362  ;
                                     2362  ; Date/time string for special SYSTEM screen
                                     2362  ;
                                     2362
                                     2362  SYS_TIME_STR::
                                 0E' 2362             .BYTE   10$-5$
                         39 01 59 1B  2363  5$:          .BYTE   ESC,^A/Y/,1,57
                            44 41 21  2367              .ASCII  \!AD\
                         45 01 59 1B  236A              .BYTE   ESC,^A/Y/,1,69
                            44 41 21  236E              .ASCII  \!AD\
                                     2371  10$:
                                     2371
                                     2371  ;
                                     2371  ; Summary line string, including cursor
                                     2371  ; positioning and from/to times.
                                     2371  ;
                                     2371
                                     2371  SUMMLINE_STR::
                                 2C' 2371             .BYTE   10$-5$
                         37 03 59 1B  2372  5$:          .BYTE   ESC,^A/Y/,3,55
          44 41 21 20 3A 6D 6F 72 46  2376              .ASCII  \From: !AD\
                         25 04 59 1B  237F              .BYTE   ESC,^A/Y/,4,37
 20 20 20 20 20 59 52 41 4D 4D 55 53  2383              .ASCII  \SUMMARY           To:   !AD\
 20 20 20 3A 6F 54 20 20 20 20 20 20  238F
                            44 41 21  239B
                                     239E  10$:
                                     239E
                                     239E  ;
                                     239E  ; Special summary line string for SYSTEM class
                                     239E  ;
                                     239E
                                     239E  SYS_SUMMLINE_STR::
                                 25' 239E             .BYTE   10$-5$
                         37 01 59 1B  239F  5$:          .BYTE   ESC,^A/Y/,1,55
          44 41 21 20 3A 6D 6F 72 46  23A3              .ASCII  \From: !AD\
                         37 02 59 1B  23AC              .BYTE   ESC,^A/Y/,2,55
          44 41 21 20 20 20 3A 6F 54  23B0              .ASCII  \To:   !AD\
                         25 03 59 1B  23B9              .BYTE   ESC,^A/Y/,3,37
                59 52 41 4D 4D 55 53  23BD              .ASCII  \SUMMARY\
                                     23C4  10$:
                                     23C4
                                     23C4  PLAY_STR:                                   ; String for footing line
                                 0E' 23C4             .BYTE   10$-5$
                               52 1B  23C5  5$:          .BYTE   ESC,^A/R/                 ; Reverse video
       20 4B 43 41 42 59 41 4C 50 20  23C7              .ASCII  \ PLAYBACK \
                          55 1B  23D1    624            .BYTE   ESC,^A/U/               ; Undo the reverse
                                 23D3    625 10$:
                                 23D3    626
                                 23D3    627 SUMM_STR:                              ; String for footing line
                            11'  23D3    628            .BYTE   10$-5$
                          52 1B  23D4    629 5$:        .BYTE   ESC,^A/R/               ; Reverse video
47 4E 49 5A 49 52 41 4D 4D 55 53 20  23D6    630            .ASCII  \ SUMMARIZING \
                             20  23E2
                          55 1B  23E3    631            .BYTE   ESC,^A/U/               ; Undo the reverse
                                 23E5    632 10$:
                                 23E5    633
                                 23E5    634 REC_STR:                               ; String for footing line
                            0F'  23E5    635            .BYTE   10$-5$
                          52 1B  23E6    636 5$:        .BYTE   ESC,^A/R/               ; Reverse video
   20 47 4E 49 44 52 4F 43 45 52 20  23E8    637            .ASCII  \ RECORDING \
                          55 1B  23F3    638            .BYTE   ESC,^A/U/               ; Undo the reverse
                                 23F5    639 10$:
                                 23F5    640
                         20 00'  23F5    641 BLANK_STR::    .ASCIC  \ \               ; Blank string for footing and heading lines
                             01  23F5
                                 23F7    642 STATUS_PARMS::                         ; Status parms -- next 3 longwords
                       000023FB  23F7    643 FOOTP:         .BLKL   1                 Address of "playback" or blank string
                       000023FF  23FB    644 FOOTS:         .BLKL   1               ; Address of "summary" or blank string
                       00002403  23FF    645 FOOTR:         .BLKL   1               ; Address of "record" or blank string
```

```
                                   2403     ;
                                   2403     ; PROCESSES screen heading string.
                                   2403     ;
                                   2403
                                   2403     PROCHEAD_STR::
                               75' 2403             .BYTE   10$-5$
                       01 01 59 1B 2404     5$:     .BYTE   ESC,^A/Y/,1,1           ; Position cursor
6E 75 6F 43 20 73 73 65 63 6F 72 50 2408            .ASCII  \Process Count:\
                             3A 74 2414
                       3C 01 59 1B 2416             .BYTE   ESC,^A/Y/,1,60          ; Position cursor
                 3A 65 6D 69 74 70 55 241A          .ASCII  \Uptime:\
                       01 06 59 1B 2421             .BYTE   ESC,^A/Y/,6,1           ; Position cursor
                             4C 1B 2425             .BYTE   ESC,^A/L/               ; Underline
20 44 49 50 20 20 20 20 20 20 20 20 2427            .ASCII  \        PID   STATE PRI   NAME         PAGES\
20 49 52 50 20 45 54 41 54 53 20 20 2433
20 20 20 20 20 20 45 4D 41 4E 20 20 243F
             53 45 47 41 50 20 20 20 244B
54 4E 43 4F 49 44 20 20 20 20 20 20 2453            .ASCII  \      DIOCNT  FAULTS  CPU TIME      \
50 43 20 20 53 54 4C 55 41 46 20 20 245F
20 20 20 20 20 20 45 4D 49 54 20 55 246B
                             55 1B 2477             .BYTE   ESC,^A/U/               ; Undo underlining
                                   2479     10$:
                                   2479
                                   2479     ;
                                   2479     ; Tabular screen heading string.
                                   2479     ;
                                   2479
                                   2479     TABHEAD_STR::
                               43' 2479             .BYTE   10$-5$
                             4C 1B 247A     5$:     .BYTE   ESC,^A/L/               ; Underline
                          52 55 43 247C             .ASCII  \CUR\
                             55 1B 247F             .BYTE   ESC,^A/U/               ; Undo underlining
   20 20 20 20 20 20 20 43 41 21 2481               .ASCII  \!AC       \
                             4C 1B 248B             .BYTE   ESC,^A/L/               ; Underline
                          45 56 41 248D             .ASCII  \AVE\
                             55 1B 2490             .BYTE   ESC,^A/U/               ; Undo underlining
20 20 20 20 20 20 20 43 41 21 2D 21 2492            .ASCII  \!-!AC       \
                             4C 1B 249E             .BYTE   ESC,^A/L/               ; Underline
                          4E 49 4D 24A0             .ASCII  \MIN\
                             55 1B 24A3             .BYTE   ESC,^A/U/               ; Undo underlining
20 20 20 20 20 20 20 43 41 21 2D 21 24A5            .ASCII  \!-!AC       \
                             4C 1B 24B1             .BYTE   ESC,^A/L/               ; Underline
                          58 41 4D 24B3             .ASCII  \MAX\
                             55 1B 24B6             .BYTE   ESC,^A/U/               ; Undo underlining
                    43 41 21 2D 21 24B8             .ASCII  \!-!AC\
                                   24BD     10$:
                                     24BD     688 ;
                                     24BD     689 ; Heading string for special SYSTEM class screen
                                     24BD     690 ;
                                     24BD     691
                                     24BD     692 SYS_HEAD_STR::
                                 1F' 24BD     693         .BYTE   10$-5$
                         01 01 59 1B 24BE     694 5$:     .BYTE   ESC,^A/Y/,1,1
             43 41 21 20 3A 65 64 6F 4E 24C2  695         .ASCII  \Node: !AC\
                         01 02 59 1B 24CB     696         .BYTE   ESC,^A/Y/,2,1
       20 3A 63 69 74 73 69 74 61 74 53 24CF  697         .ASCII  \Statistic: \
                            44 41 21 24DA     698         .ASCII  \!AD\
                                     24DD     699 10$:
                                     24DD     700
                                     24DD     701 ;
                                     24DD     702 ; Bar graph screen heading string.
                                     24DD     703 ;
                                     24DD     704
                                     24DD     705 BARHEAD_STR::
                                 40' 24DD     706         .BYTE   10$-5$
                         26 06 59 1B 24DE     707 5$:     .BYTE   ESC,^A/Y/,6,38
 3E 21 43 41 21 4C 55 21 3C 30 31 21 24E2     708         .ASCII  \!10<!UL!AC!>!10<!UL!AC!>!10<!UL!AC!>!6<!UL!AC!>!#< !>!5UL!AC\
 3E 21 43 41 21 4C 55 21 3C 30 31 21 24EE
 3E 21 43 41 21 4C 55 21 3C 30 31 21 24FA
 21 3E 21 43 41 21 4C 55 21 3C 36 21 2506
 43 41 21 4C 55 35 21 3E 21 20 3C 23 2512
                                     251E     709 10$:
                                     251E     710
                                     251E     711 ;
                                     251E     712 ; Bar graph statistic heading string (in smaller box).
                                     251E     713 ;
                                     251E     714
                                     251E     715 STATHEAD_STR::
                                 21' 251E     716         .BYTE   10$-5$
                         0D 02 59 1B 251F     717 5$:     .BYTE   ESC,^A/Y/,2,13
                2B 2D 2D 2D 2D 2D 2B 2523     718         .ASCII  \+-----+\
                         0D 03 59 1B 252A     719         .BYTE   ESC,^A/Y/,3,13
                               20 7C 252E     720         .ASCII  \! \
                            44 41 21 2530     721         .ASCII  \!AD\
                               7C 20 2533     722         .ASCII  \ !\
                         0D 04 59 1B 2535     723         .BYTE   ESC,^A/Y/,4,13
                2B 2D 2D 2D 2D 2D 2B 2539     724         .ASCII  \+-----+\
                                     2540     725 10$:
                                     2540     726
                                     2540     727 ;
                                     2540     728 ; Other bar graph strings
                                     2540     729 ;
                                     2540     730
                               59 1B 2540     731 CURSOR_STR::    .BYTE   ESC,^A\Y\        ; Position cursor escape sequence
                                     2542     732
 20 2B 20 2D 20 2D 20 2D 20 2D 20 2B 2542     733 HORIZ_STR::     .ASCII  \+ - - - - + - - - - - + - - - - - + - - - - -+\
 20 2D 20 2B 20 2D 20 2D 20 2D 20 2D 254E
 20 2D 20 2D 20 2B 20 2D 20 2D 20 2D 255A
                2B 2D 20 2D 20 2D 2566
                                     256C     734                                          ; Top and bottom line of bar graph box
                              25 00' 256C     735 PCENT_STR::     .ASCIC  \%\              ; Percent symbol string for heading line
                                 01  256C
                              4B 00' 256E     736 K_STR::         .ASCIC  \K\              ; K symbol string for heading on bar graph b
```

```
                        01   256E
                  52 55 43   2570     737 STAT_HEAD::     .ASCII  \CUR\           ; Table of statistic headings for bar graph
                  45 56 41   2573     738                 .ASCII  \AVE\
                  4E 49 4D   2576     739                 .ASCII  \MIN\
                  58 41 4D   2579     740                 .ASCII  \MAX\
      54 4E 45 52 52 55 43   257C     741 STAT_LONG::     .ASCII   CURRENT\       ; Long version for SYSTEM class screen
      45 47 41 52 45 56 41   2583     742                 .ASCII  \AVERAGE\
      4D 55 4D 49 4E 49 4D   258A     743                 .ASCII  \MINIMUM\
      4D 55 4D 49 58 41 4D   2591     744                 .ASCII  \MAXIMUM\
                                     2598   746 :
                                     2598   747 : FAO control strings for name string displays for homogeneous classes.
                                     2598   748 :
                                     2598   749
43 41 21 3C 23 21 000025A0'010E0000' 2598   750 DISK_FAO:       .ASCID  \!#<!AC!#*$!AC!ZW:!>!#* !AF\
 21 3A 57 5A 21 43 41 21 24 2A 23 21 25A6
             46 41 21 20 2A 23 21 3E 25B2
                                     25BA   751                                        ; Disk FAO string (no alloc class)
21 24 3C 33 31 21 000025C2'010E0000' 25BA   752 DISK_FAO_AC:    .ASCID  \!13<$!ZB$!AC!ZW: !>!#* !#<(!AC)!>!AF\
 21 20 3A 57 5A 21 43 41 21 24 42 5A 25C8
 43 41 21 28 3C 23 21 20 2A 23 21 3E 25D4
                   46 41 21 3E 21 29 25E0
                                     25E6   753                                        ; Disk class FAO string (with alloc cls)
         43 41 21 000025EE'010E0000' 25E6   754 SCS_FAO::       .ASCID  \!AC\          ; SCS class FAO control string
64 6F 4E 20 6E 77 6F 6E 6B 6E 55 00' 25F1   755 UNKNOWN_NODE:   .ASCIC  /Unknown Node/ ; Counted ASCII string for a 0 length
                                  65 25FD
                                  0C 25F1
                                     25FE   756                                        ;  node name in the system block.
                                     25FE   757 :
                                     25FE   758 : FAO control string for item name display for homogeneous classes
                                     25FE   759 :
                                     25FE   760
                                     25FE   761 ITEM_NAM_STR:
                                 16' 25FE   762         .BYTE   10$-5$
                         01 00 59 1B 25FF   763 5$:     .BYTE   ESC,^A/Y/,0,1          ; Cursor position
 36 32 21 3E 21 20 20 20 20 3C 23 21 2603   764         .ASCII  \!#<    !>!26<!AC!>\
                   3E 21 43 41 21 3C 260F
                                     2615   765 10$:
                            00000006 2615   766 ILN_REG = 6                            ; Item line number for regular displays
                            00002601 2615   767 ITM[NNO = ITEM_NAM_STR+3               ; Label for item line number
```

```
                           00002616  2615   769 SYSOUT_TYPE:     .BLKB    1                   ; SYS$OUTPUT device type
                                     2616   770                                               ; MONITOR SYSOUT types are:
                                     2616   771                                               ; DEC_CRT, VT5X, OTHER_VID, HARDCOPY
                                     2616   772                                               ; (Type codes defined above)
                                     2616   773
                           00000013' 2616   774 PROC_SETUP_STR:  .LONG    20$-10$             ; Descriptor for PROCESSES screen setup ...
                           0000261E' 261A   775                  .LONG    10$                 ; ... string (incl. cursor positioning)
                                     261E   776 10$:
                         10 01 59 1B 261E   777                  .BYTE    ESC,^A/Y/,1,16      ; Cursor position to process count field
             3E 21 4C 55 21 3C 35 21 2622   778                  .ASCII   \!5<!UL!>\
                         44 01 59 1B 262A   779                  .BYTE    ESC,^A/Y/,1,68      ; Cursor position to uptime field
                            53 41 21 262E   780                  .ASCII   \!AS\
                                     2631   781 20$:
                                     2631   782
                           0000003D' 2631   783 PROC_RES_STR:    .LONG    20$-10$             ; Descriptor for PROCESSES FAO control ...
                           00002639' 2635   784                  .LONG    10$                 ; ... string (resident process)
 41 35 21 20 4C 58 21 20 20 20 20 20 2639   785 10$:             .ASCII   \     !XL !5AC !2UL !4(+)!15AF !9<!UL/!UL!> !7UL !7UL !2(+)!
 29 2B 28 34 21 20 4C 55 32 21 20 43 2645
 4C 55 21 3C 39 21 20 46 41 35 31 21 2651
 20 4C 55 37 21 20 3E 21 4C 55 21 2F 265D
 25 21 29 2B 28 32 21 20 4C 55 37 21 2669
                                  54 2675
                                     2676   786 20$:
                                     2676   787
                           00000045' 2676   788 PROC_NRES_STR:   .LONG    20$-10$             ; Descriptor for PROCESSES FAO control ...
                           0000267E' 267A   789                  .LONG    10$                 ; ... string (non-resident process)
 41 35 21 20 4C 58 21 20 20 20 20 20 267E   790 10$:             .ASCII   \     !XL !5AC !2UL !4(+)!15AF !9<!UL/!UL!>!5(+)       SWAPP
 29 2B 28 34 21 20 4C 55 32 21 20 43 268A
 4C 55 21 3C 39 21 20 46 41 35 31 21 2696
 20 29 2B 28 35 21 3E 21 4C 55 21 2F 26A2
 45 50 50 41 57 53 20 20 20 20 20 20 26AE
          20 2A 39 21 54 55 4F 20 44 26BA
                                     26C3   791 20$:
                                     26C3   792
                                     26C3   793 ;
                                     26C3   794 ; Top PROCESSES FAO control strirg for one process display line
                                     26C3   795 ;
                                     26C3   796
                                     26C3   797 TOPSTR:
                                 2B' 26C3   798         .BYTE   10$-5$
                         02 00 59 1B 26C4   799 5$:     .3YTE   ESC,^A/Y/,0,2           ; position to left margin
 31 21 29 2B 28 34 21 20 20 4C 58 21 26C8   800         .ASCII  \!XL  !4(+)!15AF   !7<!#UL!>  \
 55 23 21 3C 37 21 20 20 20 46 41 35 26D4
                      20 20 3E 21 4C 26E0
                               46 1B 26E5   801         .BYTE   ESC,^A/F/               ; select alternate char set
                         2A 2A 23 21 26E7   802         .ASCII  \!#**\                  ; repeating bar character
                         4B 1B 47 1B 26EB   803         .BYTE   ESC,^A/G/,ESC,^A/K/     ; select reg set and erase to EOL
                                     26EF   804 10$:
                           000026C6  26EF   805 TOPLNNO = TOPSTR+3                      ; label for line number
                           000026EA  26EF   806 TOPBAR  = TOPSTR+39                     ; label for bar character
                                     26EF   807
                                     26EF   808 ;
                                     26EF   809 ; Top PROCESSES FAO control string to erase a line
                                     26EF   810 .
                                     26EF   811
                                     26EF   812 ERLINE_STR:
                                 06' 26EF   813         .BYTE   10$-5$
```

```
                 01 00 59 1B  26F0    814 5$:         .BYTE   ESC,^A/Y/,0,1             ; position to left margin
                       4B 1B  26F4    815             .BYTE   ESC,^A/K/                 ; erase to end of line
                              26F6    816 10$:
                   000026F2   26F6    817 ERLNNO = ERLINE_STR+3                         ; label for line number
                              26F6    818
                   00000004'  26F6    819 VT100_REGSET:   .LONG   20$-10$               ; descriptor for VT100 ...
                   000026FE'  26FA    820                 .LONG   10$                   ; ... "regular" char set esc seq
                 0F 42 28 1B  26FE    821 10$:            .BYTE   ESC,^A/(/,^A/B/,SI
                              2702    822 20$:
                   00000003'  2702    823 VT100_ALTSET:   .LONG   20$-10$               ; descriptor for VT100 ...
                   0000270A'  2706    824                 .LONG   10$                   ; ... "alternate" graphics set esc seq
                    30 28 1B  270A    825 10$:            .BYTE   ESC,^A/(/,^A/0/
                              270D    826 20$:
                   00002711   270D    827 VT100_CURSET:   .BLKL   1                     ; addr of esc seq descr for curr char set
                   00000002'  2711    828 REG_SET:        .LONG   20$-10$               ; VT52 esc seq to estab regular char set
                   00002719'  2715    829                 .LONG   10$
                       47 1B  2719    830 10$:            .BYTE   ESC,^A/G/
                              271B    831 20$:
                              271B    832
                              271B    833 PUTSCRAPG:                                    ; arg list for PUT_SCREEN call
                   00000004   271B    834         .LONG   4                             ; argument count
                   0000272F'  271F    835         .LONG   TXT_DESC                      ; addr of buffer to display
          00000000 00000000   2723    836         .LONG   0,0                           ; no cursor pos specification
                   00000000   272B    837 ATTRIBMSK: .LONG 0                            ; start off with no special attributes
                              272F    838
                              272F    839 TXT_DESC:                                     ; hold area for descriptor to be PUT'd
                   00002733   272F    840 TXT_LENGTH:     .BLKL   1                     ; length
                   00002737   2733    841 TXT_START:      .BLKL   1                     ; address
                              2737    842
59 55 52 4B 4A 48 47 46 42 4C 2737    843 ESC_SEQ_TABLE:  .ASCII  \LBFGHJKRUY\          ; table of valid escape modifiers
                   0000000A   2741    844 ES_TAB_LEN = .-ESC_SEQ_TABLE                  ; length of table
                              2741    845
                   000027A9   2741    846 PUTMSGVEC:      .BLKL   PUTMSGSIZE            ; Message argument vector for $PUTMSG
```

```
    27A9          .SBTTL  CALC_LEN - Calculate class record lengths
00000000          .PSECT  $$MONCODE,NOWRT,EXE
    0000    ;++
    0000    ;
    0000    ; FUNCTIONAL DESCRIPTION:
    0000    ;
    0000    ;       This routine is called to calculate the length of a block
    0000    ;       (CDB$W_BLKLEN) for each STANDARD class. Non-standard
    0000    ;       classes have this value entered at compile-time. Block
    0000    ;       length for standard heterogeneous classes is defined as the
    0000    ;       sum of the sizes of all data items which comprise the class,
    0000    ;       and which are recorded (calculated items which are displayed but
    0000    ;       not recorded are not included in block length).
    0000    ;       Block length for standard homogeneous classes is the sum of
    0000    ;       all data items comprising the class (including the element ID)
    0000    ;       for a SINGLE element (e.g., for a single disk).
    0000    ;
    0000    ;       In addition, some pre-processing for the MODES class is done,
    0000    ;       and, for homogeneous classes, the CDX$W_CUMELCT, CDX$B_IDISCONSEC
    0000    ;       CDX$W_IBITS and CDX$B_IDISCT fields are initialized.
    0000    ;
    0000    ; CALLING SEQUENCE:
    0000    ;
    0000    ;       CALLS #1,CALC_LEN
    0000    ;
    0000    ; INPUTS:
    0000    ;
    0000    ;       4(AP)   - address of MRB$O_CLASSBITS, the bit string
    0000    ;                 representing classes to be monitored.
    0000    ;
    0000    ; IMPLICIT INPUTS:
    0000    ;
    0000    ;       PERFTABLE     - table of IDB's describing each data item,
    0000    ;                       indexed by item number ( * entry size).
    0000    ;
    0000    ;       CDBHEAD       - table of CDB's, one for each class.
    0000    ;
    0000    ;       MAX_CLASS_NO - maximum class number (class numbers are zero-origin)
    0000    ;
    0000    ;       MODES_CLSNO  - MODES class number
    0000    ;
    0000    ;       MODES_ICOUNT  - MODES item count (for uniprocessor)
    0000    ;
    0000    ; OUTPUTS:
    0000    ;
    0000    ;       None
    0000    ;
    0000    ; IMPLICIT OUTPUTS:
    0000    ;
    0000    ;       CDB$W_BLKLEN (block length) field established for each CDB.
    0000    ;
    0000    ;       PROCS_PER_REC field established for the PROCESSES class.
    0000    ;
    0000    ;
    0000    ; ROUTINE VALUE:
    0000    ;
    0000    ;       R0 = SS$_NORMAL, or MNR$_ITMNOTDEF
```

```
                             0000  ;
                             0000  ; SIDE EFFECTS:
                             0000  ;
                             0000  ;         None
                             0000  ;
                             0000  ;--
                             0000
                             0000
                       0F5C  0000  .ENTRY  CALC_LEN,       ^M<R2,R3,R4,R6,R8,R9,R10,R11>
                             0002
                             0002  ;
                             0002  ; First, re-establish item count for the MODES class (uniprocessor)
                             0002  ;
                             0002
16 04 BC    00000000'8F  E1  0002          BBC     #MODES_CLSNO,@4(AP),10$          ; Skip if MODES not present
      56    00000000'EF  DE  000B          MOVAL   CDBHEAD,R6                       ; Get address of first CDB
      56    00000000'8F  C0  0012          ADDL    #<CDB$K_SIZE*MODES_CLSNO>,R6     ; Calculate addr of MODES CDB
   14 A6    00000000'8F  D0  0019          MOVL    #MODES_ICOUNT,CDB$L_ICOUNT(R6)   ; Get uniprocessor item count
                             0021
                             0021  ;
                             0021  ; Now calculate CDB$W_BLKLEN for all requested classes.
                             0021  ;
                             0021
                             0021  10$:
                  5B     D4  0021          CLRL    R11                         ; Init starting bit position
                             0023  20$:
            59    20     D0  0023          MOVL    #32,R9                      ; Init bit field size
                             0026                                              ; NOTE -- must handle in 32-bit chunks
            58    5B     D0  0026          MOVL    R11,R8                      ; Init start position of next chunk
                             0029  30$:
 5A   04 BC     59    58     EA  0029          FFS     R8,R9,@4(AP),R10            ; Search for next class number
                             002F                                              ; R10 contains class no. if found
                  11     13  002F          BEQL    40$                         ; Branch if none found this chunk
                  18     10  0031          BSBB    CALC_CLASS                  ; Calc block length for this class
               14 50     E9  0033          BLBC    R0,50$                      ; Go return if error
            59    58     C0  0036          ADDL2   R8,R9                       ; Compute next starting
      58    5A    01     C1  0039          ADDL3   #1,R10,R8                   ; ... position and field size
            59    58     C2  003D          SUBL2   R8,R9                       ; ... for this chunk
                  E7     11  0040          BRB     30$                         ; Go search rest of chunk
                             0042  40$:
FFD9 5B    20     0000'8F  3D  0042          ACBW    #MAX_CLASS_NO,#32,R11,20$ ; Loop to process next chunk
                             004A
                             004A  ;
                             004A  ; At this point, CDB$W_BLKLEN fields for all monitored classes
                             004A  ; have been established.
                             004A  ;
                             004A
                             004A  50$:
                         04  004A          RET                                 ; Return with status in R0
                             004B
                             004B
                             004B  CALC_CLASS:                                 ; Calc block length for this class
                             004B                                              ; NOTE -- R10 contains class number
                             004B                                              ; Regs R8 thru R11 must not be changed
                             004B
 56    5A    00000053 8F   C5  004B          MULL3   #CDB$K_SIZE,R10,R6          ; Compute offset to desired CDB
    56   00000000'EF46   9E  0053          MOVAB   CDBHEAD[R6],R6              ; Index to CDB address
```

```
                                 005B    962
                                 005B    963 ;
                                 005B    964 ; For all classes except homogeneous standard classes, set the
                                 005B    965 ; element count (of elements displayed) equal to the item count
                                 005B    966 ; (of items collected).
                                 005B    967 ;
                                 005B    968
          05 4B A6    05    E0   005B    969          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if homog
           18 A6   14 A6    D0   0060    970          MOVL    CDB$L_ICOUNT(R6),CDB$L_ECOUNT(R6) ; Item count = elt count
                                 0065    971 10$:
          10 4B A6    04    E0   0065    972          BBS     #CDB$V_STD,CDB$L_FLAGS(R6),20$ ; Br if a standard class
      52    00007CEB 8F     D0   006A    973          MOVL    #<MAX_REC_SIZE-MNR_CLS$K_HSIZE-MNR_PRO$K_PSIZE>,R2
                                 0071    974                                          ; Get max data size
0016'CF     52    20 A6     A7   0071    975          DIVW3   CDB$W_BLKLEN(R6),R2,W^PROCS_PER_REC
                                 0078    976                                          ; Compute processes per record for ...
                                 0078    977                                          ; ... PROCESSES non-STD class
                  54    11       0078    978          BRB     80$                     ; All done with this class
                                 007A    979 20$:
            50    14 A6     D0   007A    980          MOVL    CDB$L_ICOUNT(R6),R0     ; Get no of items to sum for this CDB
            51    1C A6     D0   007E    981          MOVL    CDB$A_ITMSTR(R6),R1     ; Address of item-number string
                        52  D4   0082    982          CLRL    R2                      . Clear block size reg
                                 0084    983 30$:
              54    81      9A   0084    984          MOVZBL  (R1)+,R4                ; Get next item number
              54    11      C4   0087    985          MULL    #IDB$K_ILENGTH,R4       ; Compute index into IDB table
       54    0000'CF44      9E   008A    986          MOVAB   W^PERFTABLE[R4],R4      ; Address of IDB for this item
                 10 A4      95   0090    987          TSTB    IDB$B_FLAGS(R4)         ; Is this a calculated item?
                    17      12   0093    988          BNEQ    70$                     ; Branch if so (don't add to size)
                                 0095    989          CASE    IDB$W_ISIZE(R4),<40$,50$,60$>,W ; Select on proper size
                                 00A0    990
                    52      D6   00A0    991 40$:     INCL    R2                      ; Add 1 for byte
                    08      11   00A2    992          BRB     70$
              52    02      C0   00A4    993 50$:     ADDL    #2,R2                   ; Add 2 for word
                    03      11   00A7    994          BRB     70$
              52    04      C0   00A9    995 60$:     ADDL    #4,R2                   ; Add 4 for longword
                 D5 50      F5   00AC    996 70$:     SOBGTR  R0,30$                  ; Loop for each item in this class
                                 00AF    997
           20 A6    52      B0   00AF    998          MOVW    R2,CDB$W_BLKLEN(R6)     ; Store away size for this class
                                 00B3    999
                                 00B3   1000 ;
                                 00B3   1001 ; Now add in size of element ID for homogeneous classes
                                 00B3   1002 ;
                                 00B3   1003
          16 4B A6    05    E1   00B3   1004          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),80$ ; All done if hetero
            52    32 A6     D0   00B8   1005          MOVL    CDB$A_CDX(R6),R2        ; Get CDX addr for homog class
            50    09 A2     9A   00BC   1006          MOVZBL  CDX$B_ELIDLEN(R2),R0    ; Get length of element ID
           20 A6    50      A0   00C0   1007          ADDW2   R0,CDB$W_BLKLEN(R6)     ; Add it in to get data block size
                                 00C4   1008 ;
                                 00C4   1009 ; Also, for homogeneous classes, initialize CDX$W_CUMELCT
                                 00C4   1010 ; and CDX$B_IDISCONSEC, and calculate display item count.
                                 00C4   1011 ;
                                 00C4   1012
                 0A A2      B4   00C4   1013          CLRW    CDX$W_CUMELCT(R2)       ; Init cumulative element count
                 07 A2      94   00C7   1014          CLRB    CDX$B_IDISCONSEC(R2)    ; Init consecutive display number
                                 00CA   1015
                    06      10   00CA   1016          BSBB    CALC_DITEM              ; Calculate display item count
                    03      11   00CC   1017          BRB     90$                     ; Go return with status in R0
                                 00CE   1018 80$:
```

```
          50   01   D0  00CE  1019         MOVL    #SS$_NORMAL,R0          ; Success status
                        00D1  1020
                    05  00D1  1021 90$:    RSB                             ; Return with status in R0
```

```
                                 00D2  CALC_DITEM:                                   ; Calc display item cnt for homog class
                                 00D2
             50    14 A6    D0   00D2          MOVL    CDB$L_ICOUNT(R6),R0           ; Get count of all items for class
             26 62    0F    E0   00D6          BBS     #CDX$S_IBITS-1, -
                                 00DA                  CDX$W_IBITS(R2),30$           ; Br if ALL items requested for display
                                 00DA
                                 00DA  ;
                                 00DA  ; Use FFS loop to calculate the number of display items requested.
                                 00DA  ; Store the number in CDX$B_IDISCT. Also, clear any bits higher
                                 00DA  ; than the number of items defined for this revision level.
                                 00DA  ;
                                 00DA
            51    10    50    83  00DA          SUBB3   R0,#CDX$S_IBITS,R1            ; Calc number of unused bits
       62   51    50    00    F0  00DE          INSV    #0,R0,R1,CDX$W_IBITS(R2) ; ... and make sure they are clear
                        54    D4  00E3          CLRL    R4                            ; Init counter of requested items
                        51    D4  00E5          CLRL    R1                            ; Init starting bit number for FFS
                                 00E7  10$:
       53   62    50    51    EA  00E7          FFS     R1,R0,CDX$W_IBITS(R2),R3 ; Search for next item number
                                 00EC
                                 00EC                                                ; R3 contains item number if found
                        0C    13  00EC          BEQL    20$                           ; Branch if none found
                                 00EE
                        54    D6  00EE          INCL    R4                            ; Count this item
                                 00F0
            51    53    01    C1  00F0          ADDL3   #1,R3,R1                      ; Compute next starting ...
            50    10    51    C3  00F4          SUBL3   R1,#CDX$S_IBITS,R0            ; ... position and field size
                        ED    11  00F8          BRB     10$                           ; Go search rest of bit string
                                 00FA
                                 00FA  20$:
               06 A2    54    90  00FA          MOVB    R4,CDX$B_IDISCT(R2)           ; Store number of display items
                        0F    11  00FE          BRB     40$                           ; ... and go return
                                 0100
                                 0100  ;
                                 0100  ; ALL items requested for display. Store number requested in
                                 0100  ; CDX$B_IDISCT, and set all item bits in CDX$W_IBITS.
                                 0100  ;
                                 0100
                                 0100  30$:
                        62    B4  0100          CLRW    CDX$W_IBITS(R2)               ; Start out with all item bits clear
 62     50    00   FFFFFFFF 8F    F0  0102          INSV    #-1,#0,R0,CDX$W_IBITS(R2) ; Set all bits defined for this rev.
               06 A2    50    90  010B          MOVB    R0,CDX$B_IDISCT(R2)           ; ... and store its count
                                 010F  40$:
                                 010F
                  50    01    D0  010F          MOVL    #SS$_NORMAL,R0                ; Assume normal status
                        62    B5  0112          TSTW    CDX$W_IBITS(R2)               ; Check if no items requested
                        14    12  0114          BNEQU   50$                           ; Br if at least one requested
                                 0116
               00000000'8F    DD  0116          PUSHL   #MNR$_ITMNOTDEF               ; Stack MONITOR failing status code
         00001EB6'EF    01    FB  011C          CALLS   #1,MON_ERR                    ; Log the error
         50    00000000'8F    D0  0123          MOVL    #MNR$_ITMNOTDEF,R0            ; Get status to caller
                                 012A  50$:
                        05        012A          RSB                                   ; Return to caller
```

```
012B          .SBTTL  MOVE_CLASS_QUALS - Move Class Qualifier Values
012B    ;++
012B    ;
012B    ; FUNCTIONAL DESCRIPTION:
012B    ;
012B    ;       This routine is called to move a set of values from
012B    ;       one CDB or CDX field to another. In particular, the QFLAGS
012B    ;       (class qualifier flags), IBITS (item bits) and the ST
012B    ;       (display statistic) values are moved among three fields
012B    ;       defined for each, representing default value, current value
012B    ;       and active value. The types of moves are defined below under
012B    ;       INPUTS.
012B    ;
012B    ; CALLING SEQUENCE:
012B    ;
012B    ;       CALLS #1,MOVE_CLASS_QUALS
012B    ;
012B    ; INPUTS:
012B    ;
012B    ;       4(AP) - address of a byte containing a code indicating which
012B    ;               type of move to make, as follows:
012B    ;
012B    ;               If code =
012B    ;
012B    ;               DEF_TO_CUR(=0)  =>  Move default values to current values.
012B    ;
012B    ;               CUR_TO_ACT(=1)  =>  Move current values to active values.
012B    ;                                   In addition, clear the CDB$V_EXPLIC bit.
012B    ;
012B    ;               ACT_TO_CUR(=2)  =>  Move active values to current values.
012B    ;
012B    ;               ALL_TO_ACT(=3)  =>  Move the ALL statistic value to active.
012B    ;
012B    ; IMPLICIT INPUTS:
012B    ;
012B    ;       CDBHEAD      - table of CDB's, one for each class.
012B    ;
012B    ;       MAX_CLASS_NO - maximum class number (class numbers are zero-origin)
012B    ;
012B    ;       CDB$B_ST, CDB$B_ST_DEF and CDB$B_ST_CUR fields for each CDB.
012B    ;
012B    ;       CDB$W_QFLAGS, CDB$W_QFLAGS_DEF and CDB$W_QFLAGS_CUR fields for each CDB.
012B    ;
012B    ;       CDX$W_IBITS, CDX$W_IBITS_DEF and CDX$W_IBITS_CUR fields for each homog class
012B    ;
012B    ; OUTPUTS:
012B    ;
012B    ;       None
012B    ;
012B    ; IMPLICIT OUTPUTS:
012B    ;
012B    ;       Requested move is performed.
012B    ;
012B    ; ROUTINE VALUE:
012B    ;
012B    ;       R0 = SS$_NORMAL
012B    ;
```

```
                              012B  1133 ; SIDE EFFECTS:
                              012B  1134 ;
                              012B  1135 ;         None
                              012B  1136 ;
                              012B  1137 ;--
                              012B  1138
                              012B  1139
                        0048  012B  1140 .ENTRY  MOVE_CLASS_QUALS,       ^M<R3,R6>
                              012D  1141
56    00000000'EF        DE   012D  1142         MOVAL   CDBHEAD,R6              ; Get address of first CDB
53    00000001'8F        D0   0134  1143         MOVL    #MAX_CLASS_NO+1,R3      ; Get number of CDB's
                              013B  1144         CASE    @4(AP),<10$,20$,30$,40$>,B ; Select on type of move
                0090     31   0148  1145         BRW     50$                     ; Do nothing if out of range
                              014B  1146
                              014B  1147 ;
                              014B  1148 ; DEFAULT TO CURRENT
                              014B  1149 ;
                              014B  1150
                              014B  1151 10$:
   44 A6    43 A6        90   014B  1152         MOVB    CDB$B_ST_DEF(R6),CDB$B_ST_CUR(R6)             ; Load default stat to cur
   49 A6    47 A6        B0   0150  1153         MOVW    CDB$W_QFLAGS_DEF(R6),CDB$W_QFLAGS_CUR(R6) ; Load default qual flags
   09 4B A6    05        E1   0155  1154         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),15$             ; Br if heterogeneous clas
      50    32 A6        D0   015A  1155         MOVL    CDB$A_CDX(R6),R0                              ; Get CDX address
   04 A0    02 A0        B0   015E  1156         MOVW    CDX$W_IBITS_DEF(R0),CDX$W_IBITS_CUR(R0)       ; Load def item bits to cu
                              0163  1157 15$:
56    00000053 8F        C0   0163  1158         ADDL    #CDB$K_SIZE,R6                                ; Point to CDB for next cl
             DE 53       F5   016A  1159         SOBGTR  R3,10$                                        ; Loop for each CDB
              006B       31   016D  1160         BRW     50$                                           ; Go to common exit
                              0170  1161
                              0170  1162 ;
                              0170  1163 ; CURRENT TO ACTIVE
                              0170  1164 ;
                              0170  1165
                              0170  1166 20$:
   42 A6    44 A6        90   0170  1167         MOVB    CDB$B_ST_CUR(R6),CDB$B_ST(R6)                 ; Load current stat to act
   45 A6    49 A6        B0   0175  1168         MOVW    CDB$W_QFLAGS_CUR(R6),CDB$W_QFLAGS(R6)         ; Load current qual flags
   00 4B A6    0C        E5   017A  1169         BBCC    #CDB$V_EXPLIC,CDB$L_FLAGS(R6),22$             ; Indicate no explicit qua
                              017F  1170 22$:
   08 4B A6    05        E1   017F  1171         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),25$              ; Br if heterogeneous clas
      50    32 A6        D0   0184  1172         MOVL    CDB$A_CDX(R6),R0                              ; Get CDX address
      60    04 A0        B0   0188  1173         MOVW    CDX$W_IBITS_CUR(R0),CDX$W_IBITS(R0)           ; Load curr item bits to a
                              018C  1174 25$:
                              018C  1175
56    00000053 8F        C0   018C  1176         ADDL    #CDB$K_SIZE,R6          ; Point to CDB for next class
             DA 53       F5   0193  1177         SOBGTR  R3,20$                  ; Loop for each CDB
              0042       31   0196  1178         BRW     50$                     ; Go to common exit
                              0199  1179
                              0199  1180 ;
                              0199  1181 ; ACTIVE TO CURRENT
                              0199  1182 ;
                              0199  1183
                              0199  1184 30$:
   44 A6    42 A6        90   0199  1185         MOVB    CDB$B_ST(R6),CDB$B_ST_CUR(R6)                 ; Load active stat back to
   49 A6    45 A6        B0   019E  1186         MOVW    CDB$W_QFLAGS(R6),CDB$W_QFLAGS_CUR(R6)         ; Load active qual flags b
   08 4B A6    05        E1   01A3  1187         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),35$              ; Br if heterogeneous clas
      50    32 A6        D0   01A8  1188         MOVL    CDB$A_CDX(R6),R0                              ; Get CDX address
   04 A0       60        B0   01AC  1189         MOVW    CDX$W_IBITS(R0),CDX$W_IBITS_CUR(R0)           ; Load active item bits ba
```

```
                              01B0   1190 35$:
                              01B0   1191
      56    00000053 8F   C0  01B0   1192          ADDL    #CDB$K_SIZE,R6          ; Point to CDB for next class
                   DF 53  F5  01B7   1193          SOBGTR  R3,30$                  ; Loop for each CDB
                      1F  11  01BA   1194          BRB     50$                     ; Go to common exit
                              01BC   1195
                              01BC   1196 ;
                              01BC   1197 ; /ALL TO ACTIVE
                              01BC   1198 ;
                              01BC   1199
                              01BC   1200 40$:
         10 4B A6     04  E1  01BC   1201          BBC     #CDB$V_STD,CDB$L_FLAGS(R6),45$          ; Br if non-standard class
         44 A6     43 A6  91  01C1   1202          CMPB    CDB$B_ST_DEF(R6),CDB$B_ST_CUR(R6)       ; Is default stat equal to
                      09  12  01C6   1203          BNEQ    45$                                     ; Br if not
         04 4B A6     0C  E0  01C8   1204          BBS     #CDB$V_EXPLIC,CDB$L_FLAGS(R6),45$       ; Br if explicit qualifier
            42 A6     00  90  01CD   1205          MOVB    #ALL_STAT,CDB$B_ST(R6)                  ; Force the ALL statistic
                              01D1   1206 45$:
      56    00000053 8F   C0  01D1   1207          ADDL    #CDB$K_SIZE,R6          ; Point to CDB for next class
                   E1 53  F5  01D8   1208          SOBGTR  R3,40$                  ; Loop for each CDB
                              01DB   1209
                              01DB   1210 50$:
               50     01  D0  01DB   1211          MOVL    #SS$_NORMAL,R0          ; Indicate success
                          04  01DE   1212          RET                             ; ... and return
```

```
                                01DF         .SBTTL  FETCH - Collect Data into Buffer
                                01DF
                                01DF  ;++
                                01DF  ;
                                01DF  ; FUNCTIONAL DESCRIPTION:
                                01DF  ;
                                01DF  ;      This routine is called to collect the data for the next interval.
                                01DF  ;      It scans a table describing which items to collect, and moves
                                01DF  ;      each item to the proper slot in the collection buffer supplied
                                01DF  ;      by the caller.
                                01DF  ;
                                01DF  ; CALLING SEQUENCE:
                                01DF  ;
                                01DF  ;      Entered via CALL from $CMKRNL system service.
                                01DF  ;
                                01DF  ; INPUTS:
                                01DF  ;
                                01DF  ;       4(AP) - address of CDB (Class Descriptor Block)
                                01DF  ;
                                01DF  ;       8(AP) - address of 1st byte of variable portion of collection buffer
                                01DF  ;
                                01DF  ; IMPLICIT INPUTS:
                                01DF  ;
                                01DF  ;      EXE$GQ_SYSTIME - current time in system time (quadword) units
                                01DF  ;
                                01DF  ;      PERFTABLE - table describing each data item, indexed by
                                01DF  ;                      item number ( * entry size)
                                01DF  ;
                                01DF  ; OUTPUTS:
                                01DF  ;
                                01DF  ;      None
                                01DF  ;
                                01DF  ; IMPLICIT OUTPUTS:
                                01DF  ;
                                01DF  ;      CURRENT collection buffer is filled with raw data.
                                01DF  ;
                                01DF  ; ROUTINE VALUE:
                                01DF  ;
                                01DF  ;      SS$_NORMAL
                                01DF  ;
                                01DF  ; SIDE EFFECTS:
                                01DF  ;
                                01DF  ;      None
                                01DF  ;
                                01DF  ;--
                                01DF
                                01DF
                          0078  01DF  .ENTRY  FETCH,  ^M<R3,R4,R5,R6>
                                01E1
              56    04 AC  D0   01E1         MOVL    4(AP),R6               ; Load CDB pointer
              55    08 AC  D0   01E5         MOVL    8(AP),R5               ; Load addr of 1st byte of actual data
      F6 A5   00000000'GF  7D   01E9         MOVQ    G^EXE$GQ_SYSTIME,<MNR_CLS$Q_STAMP-MNR_CLS$K_HSIZE>(R5)
                                01F1                                        ; Get current time into coll buffer
                                01F1  ;
                                01F1  ; If this class has a pre-collection routine, call it.
                                01F1  ;
                    22 A6  D5   01F1         TSTL    CDB$A_PRECOLL(R6)      ; Is there a pre-collection rtn?
              0C   13  01F4   1271          BEQL    10$                       ; No -- continue
              55   DD  01F6   1272          PUSHL   R5                        ; Yes -- stack coll buffer addr
     22 B6    01   FB  01F8   1273          CALLS   #1,@CDB$A_PRECOLL(R6)     ; Call it
        58 50      E9  01FC   1274          BLBC    R0,70$                    ; If failed, get out
        46 51      E9  01FF   1275          BLBC    R1,65$                    ; If fetch is not required, skip it
                       0202   1276 10$:
  41 4B A6    04   E1  0202   1277          [illegible] #CDB$V_STD,CDB$L_FLAGS(R6),65$ ; Skip fetch if non-STD class
  3C 4B A6    05   E0  0207   1278          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),65$ ; Skip fetch if homog class
                       020C   1279                                            ;
     50    14 A6   D0  020C   1280          MOVL    CDB$L_ICOUNT(R6),R0       ; Get number of items to fetch
     51    1C A6   D0  0210   1281          MOVL    CDB$A_ITMSTR(R6),R1       ; Address of item-number string
              53   D4  0214   1282          CLRL    R3                        ; Clear loop counter
                       0216   1283 20$:
        54    81   9A  0216   1284          MOVZBL  (R1)+,R4                  ; Get next item number
        54    11   C4  0219   1285          MULL    #IDB$K_ILENGTH,R4         ; Compute index into IDB table
  54    0000'CF44  9E  021C   1286          MOVAB   W^PERFTABLE[R4],R4        ; Address of IDB for this item
           10 A4   95  0222   1287          TSTB    IDB$B_FLAGS(R4)           ; Is this a computed item?
              02   13  0225   1288          BEQL    25$                       ; no, go collect
              1B   11  0227   1289          BRB     60$                         yes, skip this item and on to the next
                       0229   1290 25$:
                       0229   1291          CASE    IDB$W_ISIZE(R4),<30$,40$,50$>,W ; Select on proper size
                       0234   1292
     85    0C B4   90  0234   1293 30$:     MOVB    @IDB$A_ADDR(R4),(R5)+     ; Collect a byte
              0A   11  0238   1294          BRB     60$
     85    0C B4   B0  023A   1295 40$:     MOVW    @IDB$A_ADDR(R4),(R5)+     ; Collect a word
              04   11  023E   1296          BRB     60$
     85    0C B4   D0  0240   1297 50$:     MOVL    @IDB$A_ADDR(R4),(R5)+     ; Collect a longword
     CE 53    50   F2  0244   1298 60$:     AOBLSS  R0,R3,20$                 ; Loop until done
                       0248   1299
                       0248   1300 ;
                       0248   1301 ; If this class has a post-collection routine, call it.
                       0248   1302 ;
                       0248   1303
                       0248   1304 65$:
        50    01   D0  0248   1305          MOVL    #SS$_NORMAL,R0            ; Assume good status at this point
           26 A6   D5  024B   1306          TSTL    CDB$A_POSTCOLL(R6)        ; Is there a post-collection rtn?
              07   13  024E   1307          BEQL    70$                       ; No -- go return
           08 AC   DD  0250   1308          PUSHL   8(AP)                     ; Yes -- stack coll buffer addr
     26 B6    01   FB  0253   1309          CALLS   #1,@CDB$A_POSTCOLL(R6)    ; Call it
                       0257   1310
                       0257   1311 70$:
                   04  0257   1312          RET                               ; Return with status
```

```
0258            .SBTTL  CLASS_COLLECT - Collect & Transform Data
0258
0258    ;++
0258    ;
0258    ; FUNCTIONAL DESCRIPTION:
0258    ;
0258    ;       This routine is called once per interval [illegible]lass to collect
0258    ;       and record raw data and to do statistical [illegible]sformations of
0258    ;       that data. The transformations include ca[illegible]ions of minimum
0258    ;       value, maximum value, total since request [illegible]n, percent, etc.
0258    ;       On the first call to this routine per request, up to ten buffers
0258    ;       are obtained. These consist of two flip-flopped collection
0258    ;       buffers for raw data, and up to 8 statistics buffers. On subsequent
0258    ;       calls, the buffers are updated.
0258    ;
0258    ; CALLING SEQUENCE:
0258    ;
0258    ;       CALLS #1, CLASS_COLLECT
0258    ;
0258    ; INPUTS:
0258    ;
0258    ;        4(AP) - address of byte containing class number
0258    ;
0258    ; IMPLICIT INPUTS:
0258    ;
0258    ;       CDBPTR - pointer to CDB (Class Descriptor Block)
0258    ;       MRBPTR - pointer to MRB (Monitor Request Block)
0258    ;       MCAPTR - pointer to MCA (Monitor Communication Area)
0258    ;       SPTR -   pointer to SYI (System Information Area)
0258    ;
0258    ; OUTPUTS:
0258    ;
0258    ;       None
0258    ;
0258    ; IMPLICIT OUTPUTS:
0258    ;
0258    ;       Collection buffer filled with raw data for this class
0258    ;       for this interval.
0258    ;
0258    ;       All required statistics buffers filled with transformed
0258    ;       data for this class for this interval.
0258    ;
0258    ;       COLLENDED bit set to YES if this collection
0258    ;       has passed the requested ending time.
0258    ;
0258    ; ROUTINE VALUE:
0258    ;
0258    ;       R0 = NORMAL
0258    ;
0258    ; SIDE EFFECTS:
0258    ;
0258    ;       None
0258    ;
0258    ; REGISTER USAGE:
0258    ;
0258    ;       R6  = CDB pointer
0258    ;       R7  = MRB pointer
```

```
                                   0258  1371 ;        R8  = CURRENT collection buffer pointer
                                   0258  1372 ;        R9  = PREVIOUS collection buffer pointer
                                   0258  1373 ;        R10 = Buffer block pointer
                                   0258  1374 ;        R11 = MCA pointer
                                   0258  1375 ;
                                   0258  1376 ;        Others are volatile.
                                   0258  1377 ;
                                   0258  1378 ;--
                                   0258  1379
                                   0258  1380
                             0FFC  0258  1381 .ENTRY  CLASS_COLLECT,  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
              56   00000000'EF  D0  025A  1382         MOVL    CDBPTR,R6               ; Load CDB pointer
              57   00000000'EF  D0  0261  1383         MOVL    MRBPTR,R7               ; Load MRB pointer
              5B   00000000'EF  D0  0268  1384         MOVL    MCAPTR,R11              ; Load MCA pointer
                                   026F  1385
                         0C AB  D5  026F  1386         TSTL    MCA$L_COLLCNT(R11)      ; First collection ?
                            0C  12  0272  1387         BNEQ    5$                      ; No -- keep going
                   00000000'EF  16  0274  1388         JSB     GET_BUFFERS             ; Get collection & stat buffers
                         03 50  E8  027A  1389         BLBS    R0,5$                   ; Continue if OK
                          016F  31  027D  1390         BRW     CC_ERROR                ; Else exit with error
                                   0280  1391 5$:
                    5A   2E A6  D0  0280  1392         MOVL    CDB$A_BUFFERS(R6),R10   ; Load address of buffer block
              03 4B A6      05  E1  0284  1393         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if not homog class
                    5A      6A  D0  0289  1394         MOVL    (R10),R10               ; Get MBP ptr for homog class
                                   028C  1395
                                   028C  1396 ;
                                   028C  1397 ; For standard classes:
                                   028C  1398 ;
                                   028C  1399 ; Load CURRENT buffer pointer (R8) and PREVIOUS buffer pointer (R9); at
                                   028C  1400 ; the same time, flip-flop the state of the CDB$V_SWAPBUF bit, so BUFFERA
                                   028C  1401 ; and BUFFERB will be swapped on next CLASS_COLLECT call for this class.
                                   028C  1402 ; When CDB$V_SWAPBUF is OFF, BUFFERA is CURRENT; otherwise, BUFFERB is
                                   028C  1403 ; CURRENT.
                                   028C  1404 ;
                                   028C  1405 ; For non-standard class (PROCESSES):
                                   028C  1406 ;
                                   028C  1407 ; CURRENT buffer pointer (R8) always points to BUFFERA. The CDB$V_SWAPBUF
                                   028C  1408 ; bit is always clear.
                                   028C  1409 ;
                                   028C  1410
                                   028C  1411 10$:
              18 4B A6      04  E0  028C  1412         BBS     #CDB$V_STD,CDB$L_FLAGS(R6),15$ ; Branch if STD class
              01      0C AB  D1  0291  1413         CMPL    MCA$L_COLLCNT(R11),#1   ; Second collection coming up?
                            26  12  0295  1414         BNEQ    20$                     ; No -- continue
              21 43 A7      02  E1  0297  1415         BBC     #MRB$V_SUMMARY,MRB$W_FLAGS(R7),20$ ; Continue if not summarizing
                                   029C  1416 ;
                                   029C  1417 ; Swap MBP$A_BUFFA and MBP$A_BUFF1ST pointers in order to retain data
                                   029C  1418 ; from the first collection buffer for use later during summary processing.
                                   029C  1419 ;
                                   029C  1420
                    51      6A  D0  029C  1421         MOVL    MBP$A_BUFFA(R10),R1     ; Save current coll buff ptr
                 6A   04 AA  D0  029F  1422         MOVL    MBP$A_BUFF1ST(R10),MBP$A_BUFFA(R10) ; Point current to first
                 04 AA      51  D0  02A3  1423         MOVL    R1,MBP$A_BUFF1ST(R10)   ; ... and first to current
                            14  11  02A7  1424         BRB     20$                     ; Go make BufferA CURRENT
                                   02A9  1425 ;
                                   02A9  1426 ; Standard classes
                                   02A9  1427 ;
```

```
                              02A9  1428
                              02A9  1429 15$:
      0F 4B A6     01     F4  02A9  1430          BBSC    #CDB$V_SWAPBUF,CDB$L_FLAGS(R6),20$ ; Clear bit if set & branch
4B A6  01    01    01     F0  02AE  1431          INSV    #1,#CDB$V_SWAPBUF,#1,CDB$L_FLAGS(R6) ; Bit was clear -- set it
        58   04    AA     D0  02B4  1432          MOVL    MBP$A_BUFFERB(R10),R8    ; Make BufferB CURRENT
           59      6A     D0  02B8  1433          MOVL    MBP$A_BUFFERA(R10),R9    ; ... and BufferA PREVIOUS
                   07     11  02BB  1434          BRB     30$                      ; ... and continue
                              02BD  1435 20$:
           58      6A     D0  02BD  1436          MOVL    MBP$A_BUFFERA(R10),R8    ; Make BufferA CURRENT
        59   04    AA     D0  02C0  1437          MOVL    MBP$A_BUFFERB(R10),R9    ; ... and BufferB PREVIOUS
```

```
                                02C4      ;
                                02C4      ; Collect data for this class into the CURRENT collection buffer.
                                02C4      ;
                                02C4
                                02C4      30$:
             0439       30      02C4                   BSBW    COLLECTION               ; Get data for this class
            03 50       E8      02C7                   BLBS    R0,40$                   ; Continue if OK
             0122       31      02CA                   BRW     CC_ERROR                 ; Else exit with error
                                02CD      ;
                                02CD      ; If a record request, perform recording.
                                02CD      ;
                                02CD
                                02CD      40$:
    44 43 A7    01      E1      02CD                   BBC     #MRB$V_RECORD,MRB$W_FLAGS(R7),70$ ; Continue if not recording
    05 43 A7    03      E1      02D2                   BBC     #MRB$V_PLAYBACK,MRB$W_FLAGS(R7),50$ ; If live, go record
    3A 32 AB    02      E1      02D7                   BBC     #MCA$V_MULTFND,MCA$W_FLAGS(R11),70$ ; Skip rec if mult not found
                                02DC      50$:
    13 4B A6    04      E0      02DC                   BBS     #CDB$V_STD,CDB$L_FLAGS(R6),60$ ; If STD class, go write a record
       7E    20 A6      3C      02E1                   MOVZWL  CDB$W_BLKLEN(R6),-(SP)   ; Non-STD class -- push data block size
                58      DD      02E5                   PUSHL   R8                       ; ... and collection buffer ptr
  0000048A'EF   02      FB      02E7                   CALLS   #2,WRITE_PROC_RECORDS    ; Write the required num of PROCESSES recs
            25 50       E8      02EE                   BLBS    R0,70$                   ; Continue if status OK
             00FB       31      02F1                   BRW     CC_ERROR                 ; Else exit with error
                                02F4      60$:
                7E      7C      02F4                   CLRQ    -(SP)                    ; Get descr on stack for CALL
       6E    20 A6      B0      02F6                   MOVW    CDB$W_BLKLEN(R6),(SP)    ; Move in length of buffer
    07 4B A6    05      E1      02FA                   BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),65$ ; Br if a heterogeneous class
       6E    0D A8      C4      02FF                   MULL2   <MNR_CLS$K_HSIZE+MNR_HOM$L_ELTCT>(R8),(SP)
                                0303                                                    ; Times number of elts for homog class
          6E    08      C0      0303                   ADDL2   #MNR_HOM$K_PSIZE,(SP)    ; ... plus the prefix
                                0306      65$:
          6E    0D      C0      0306                   ADDL2   #MNR_CLS$K_HSIZE,(SP)    ; Add in class header size
       04 AE    58      D0      0309                   MOVL    R8,4(SP)                 ; Load address of buffer
                5E      DD      030D                   PUSHL   SP                       ; Stack descriptor address
  00000000'EF   01      FB      030F                   CALLS   #1,WRITE_RECORD          ; ... and record the buffer
                                0316
                                0316      70$:
    00'8F    04 BC      91      0316                   CMPB    @4(AP),#MODES_CLSNO      ; Is this the modes class?
                1F      12      031B                   BNEQ    80$                      ; no, branch
                                031D      ;
                                031D      ; Combine MODES counters if required
                                031D      ;
    03 4B A6    03      E1      031D                   BBC     #CDB$V_CPU_COMB,CDB$L_FLAGS(R6),75$
                                0322                                                    ; Br if not special MODES case
             00FE       30      0322                   BSBW    COMBINE_MODES            ; Combine modes for all cpu's
                                0325
                                0325
                                0325      ;
                                0325      ; Sum the first six counters to get sum of all CPU modes
                                0325      ;
                                0325      75$:
      00000000'EF       D4      0325                   CLRL    CPU_BUSY                 ; Zero CPU_BUSY
          50    06      D0      032B                   MOVL    #6,R0                    ; Get number of modes for display
    51    58    0D      C1      032E                   ADDL3   #MNR_CLS$K_HSIZE,R8,R1   ; Compute start addr of 1st set of counters
                                0332      77$:
  00000000'EF   81      C0      0332                   ADDL2   (R1)+,CPU_BUSY           ; Sum of non-idle mode counters
            F6 50       F5      0339                   SOBGTR  R0,77$                   ;
```

```
                                    033C
                                    033C
                                    033C      ;
                                    033C      ; Fill STATS buffer for homogeneous class
                                    033C      ;
                                    033C
                                    033C      80$:
          7F 4B A6    04      E1    033C                BBC     #CDB$V_STD,CDB$L_FLAGS(R6),120$ ; If non-STD class, skip all transfo
                                    0341
          11 4B A6    05      E1    0341                BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),90$ ; Br if a heterogeneous class
                      59      DD    0346                PUSHL   R9                          ; Stack addr of PREV coll buff
                      58      DD    0348                PUSHL   R8                          ; Stack addr of CURR coll buff
          00000000'EF 02      FB    034A                CALLS   #2,FILL_HOMOG_STATS         ; Fill STATS buffers for homog class
                   03 50      E8    0351                BLBS    R0,90$                      ; Continue if OK
                    0098      31    0354                BRW     CC_ERROR                    ; Else exit with error
                                    0357      90$:
                                    0357
                   0C AB      D5    0357                TSTL    MCA$L_COLLCNT(R11)          ; First collection?
                      64      13    035A                BEQL    120$                        ; Yes -- skip all transforms
                                    035C
                                    035C      ;
                                    035C      ; Calculate MCA$L_INTTICKS (clock ticks during interval just finished)
                                    035C      ;
                                    035C
            52     03 A8      7D    035C                MOVQ    MNR_CLS$Q_STAMP(R8),R2      ; Current system time to temp regs
            52     03 A9      C2    0360                SUBL2   MNR_CLS$Q_STAMP(R9),R2      ; Calc low-order in sys units
            53     07 A9      D9    0364                SBWC    MNR_CLS$Q_STAMP+4(R9),R3    ; Calc high-order in sys units
52   08 AB   52   000186A0 8F 7B    0368                EDIV    #100000,R2,MCA$L_INTTICKS(R11),R2 ; Calc interval ticks (10ms units)
                                    0372                                                                ; ... for use later
                                    0372
                                    0372      ;
                                    0372      ; Do Data Transformations for STANDARD (homogeneous) classes
                                    0372      ;
                                    0372
          43 4B A6    05      E1    0372                BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),110$ ; Br if a heterogeneous class
                                    0377
                   18 A6      D5    0377                TSTL    CDB$L_ECOUNT(R6)            ; Any elements in STATS?
                      44      13    037A                BEQL    120$                        ; No -- skip transformations
                                    037C
               51     10      D0    037C                MOVL    #CDX$S_IBITS,R1             ; Init bit field size
                      50      D4    037F                CLRL    R0                          ; Init start position
                      54      D4    0381                CLRL    R4                          ; Init item index
                                    0383      100$:
                                    0383
            52     32 A6      D0    0383                MOVL    CDB$A_CDX(R6),R2            ; Get CDB extension for HOMOG class
      53   62   51    50      EA    0387                FFS     R0,R1,CDX$W_IBITS(R2),R3    ; Search for next item number
                                    038C                                                    ; R3 contains item number if found
                      32      13    038C                BEQL    120$                        ; Branch if no more items
                                    038E
          50     1C B643      9A    038E                MOVZBL  @CDB$A_ITMSTR(R6)[R3],R0    ; Load IDB item number
               50     11      C4    0393                MULL2   #IDB$K_ILENGTH,R0           ; Compute index into IDB table
         50    0000'CF40      9E    0396                MOVAB   W^PERFTABLE[R0],R0          ; Address of IDB for this item
       00E2'CF    0A A0       B0    039C                MOVW    IDB$W_TYPE(R0),W^HOMOG_TYPE ; Save item type for COMPUTE_STATS
                                    03A2
          5A     2E B644      D0    03A2                MOVL    @CDB$A_BUFFERS(R6)[R4],R10  ; Load MBP ptr for this item
                      18      BB    03A7                PUSHR   #^M<R3,R4>                  ; Save regs
                    0044      30    03A9                BSBW    TRANSFORMS                  ; Fill trans'n buffers for this item
               18   BA  03AC              POPR    #^M<R3,R4>               ; Restore regs
                        03AE
   50    53    01   C1  03AE              ADDL3   #1,R3,R0                 ; Compute next starting ...
   51    10    50   C3  03B2              SUBL3   R0,#CDX$S_IBITS,R1       ; ... position and field size
               54   D6  03B6              INCL    R4                       ; Update item index
               C9   11  03B8              BRB     100$                     ; Go search rest of bit string
                        03BA
                        03BA       ;
                        03BA       ; Fill STATS and do Data Transformations for STANDARD (heterogeneous) classes
                        03BA       ;
                        03BA
                        03BA       110$:
             042C   30  03BA              BSBW    FILL_HETERO_STATS        ; Fill STATS from the 2 coll buffers
             0030   30  03BD              BSBW    TRANSFORMS               ; ... and fill all transformation buffs
```

```
                             03C0   1568 ;
                             03C0   1569 ; Call COLLECTION_END if end of collection has been reached. The end
                             03C0   1570 ; of collection is tested by comparing the system time (quadword)
                             03C0   1571 ; values of the current time and the requested end time.
                             03C0   1572 ;
                             03C0   1573
                             03C0   1574 120$:
    31 AB    04 BC     91    03C0   1575          CMPB     @4(AP),MCA$B_LASTC(R11)                  ; Last class?
                21     12    03C5   1576          BNEQU    CC_NORMAL                                ; No -- just exit
    28 AB    03 A8     7D    03C7   1577          MOVQ     MNR_CLS$Q_STAMP(R8),MCA$Q_LASTCOLL(R11) ; Yes -- remember latest col
    17 43 A7    04     E0    03CC   1578          BBS      #MRB$V_INDEFEND,MRB$W_FLAGS(R7),CC_NORMAL ; Skip end check if indef
    0C A7    07 A8     D1    03D1   1579          CMPL     MNR_CLS$Q_STAMP+4(R8),MRB$Q_ENDING+4(R7) ; Has curr time passed
                             03D6   1580                                                           ; ... requested end (hi-orde
                10     1F    03D6   1581          BLSSU    CC_NORMAL                                ; No -- simply return
                07     1A    03D8   1582          BGTRU    130$                                     ; Yes -- indicate so and ret
    08 A7    03 A8     D1    03DA   1583          CMPL     MNR_CLS$Q_STAMP(R8),MRB$Q_ENDING(R7)     ; Check low order longword
                07     1F    03DF   1584          BLSSU    CC_NORMAL                                ; Not at end yet -- return
                             03E1   1585 130$:
 00000000'EF    00     FB    03E1   1586          CALLS    #0,COLLECTION_END                        ; Indicate collection has en
                             03E8   1587                                                           ; COLLECTION_END sets COLLEN
                             03E8   1588
                             03E8   1589 CC_NORMAL:
 50    00000000'EF     D0    03E8   1590          MOVL     NORMAL,R0                                ; Indicate normal status
                             03EF   1591 CC_ERROR:
                       04    03EF   1592          RET                                               ; Return
```

```
                     03F0          .SBTTL  TRANSFORMS - Perform Data Transformations
                     03F0
                     03F0   ;++
                     03F0   ;
                     03F0   ; FUNCTIONAL DESCRIPTION:
                     03F0   ;
                     03F0   ;       This routine updates all transformation buffers (STATS, MIN,
                     03F0   ;       MAX, SUM, PCSTATS, PCMIN, PCMAX, PCSUM) for STANDARD classes
                     03F0   ;       (both homogeneous and heterogeneous).
                     03F0   ;
                     03F0   ; CALLING SEQUENCE:
                     03F0   ;
                     03F0   ;       BSBW    TRANSFORMS
                     03F0   ;
                     03F0   ; INPUTS:
                     03F0   ;
                     03F0   ;       R6  - CDB pointer
                     03F0   ;       R7  - MRB pointer
                     03F0   ;       R8  - CURRENT buffer pointer
                     03F0   ;       R9  - PREVIOUS buffer pointer
                     03F0   ;       R10 - Buffer block (MBP) pointer
                     03F0   ;       R11 - MCA pointer
                     03F0   ;
                     03F0   ; IMPLICIT INPUTS:
                     03F0   ;
                     03F0   ;       All transformation buffers
                     03F0   ;
                     03F0   ; OUTPUTS:
                     03F0   ;
                     03F0   ;       None
                     03F0   ;
                     03F0   ; IMPLICIT OUTPUTS:
                     03F0   ;
                     03F0   ;       All transformation buffers updated with statistics
                     03F0   ;       from the current collection.
                     03F0   ;
                     03F0   ; ROUTINE VALUE:
                     03F0   ;
                     03F0   ;       None
                     03F0   ;
                     03F0   ; SIDE EFFECTS:
                     03F0   ;
                     03F0   ;       Registers R0,R1,R2,R3,R4,R5 destroyed.
                     03F0   ;
                     03F0   ;--
                     03F0
                     03F0   TRANSFORMS:                             ; Perform data transformations
                     03F0
                     03F0   ;
                     03F0   ; Update SUM buffer from STATS buffer
                     03F0   ;
                     03F0
   50    08 AA   D0  03F0          MOVL    MBP$A_STATS(R10),R0     ; Load STATS buffer pointer
   51    14 AA   D0  03F4          MOVL    MBP$A_SUM(R10),R1       ; Load SUM buffer pointer
   52    18 A6   D0  03F8          MOVL    CDB$L_ECOUNT(R6),R2     ; Load count of elements in STATS
                     03FC   10$:
      81    80   C0  03FC          ADDL2   (R0)+,(R1)+             ; Add this item to SUM buff
```

```
                  FA 52    F5  03FF   1651          SOBGTR  R2,10$                      ; Loop for each item in STATS buff
                               0402   1652
            18 45 A6   00  E1  0402   1653          BBC     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),30$ ; If percent not requested, skip
                               0407   1654
                  049E     30  0407   1655          BSBW    FILL_PCSTATS_BUFF           ; Fill PCSTATS from STATS
                               040A   1656
                               040A   1657 ;
                               040A   1658 ; Update PCSUM buffer from PCSTATS buffer
                               040A   1659 ;
               50    18 AA  D0  040A   1660          MOVL    MBP$A_PCSTATS(R10),R0       ; Load PCSTATS buffer pointer
               51    24 AA  D0  040E   1661          MOVL    MBP$A_PCSUM(R10),R1         ; Load PCSUM buffer pointer
               52    18 A6  D0  0412   1662          MOVL    CDB$L_ECOUNT(R6),R2         ; Load count of elements in PCSTATS
                               0416   1663 20$:
                  81   80  C0  0416   1664          ADDL2   (R0)+,(R1)+                 ; Add this item to PCSUM buff
                  FA 52    F5  0419   1665          SOBGTR  R2,20$                      ; Loop for each item in PCSTATS buff
                               041C   1666 ;
                               041C   1667 ; Update PCMIN and PCMAX buffers from PCSTATS buffer
                               041C   1668 ;
                  0552     30  041C   1669          BSBW    UPD_PC_MIN_MAX              ; Update PCMIN and PCMAX
                               041F   1670
                               041F   1671 ;
                               041F   1672 ; Convert counts to rates in STATS buffer and update MIN and MAX
                               041F   1673 ;
                               041F   1674
                               041F   1675 30$:
                  04BD     30  041F   1676          BSBW    COMPUTE_STATS               ; Convert counts to rates in STATS
                               0422   1677                                              ; ... and update MIN and MAX
                               0422   1678
                           05  0422   1679          RSB                                 ; Return from TRANSFORMS subroutine
```

```
                          0423          .SBTTL  COMBINE_MODES - Combine Modes for all CPUs
                          0423
                          0423  ;++
                          0423  ;
                          0423  ; FUNCTIONAL DESCRIPTION:
                          0423  ;
                          0423  ;       This routine is called by CLASS_COLLECT to combine the mode
                          0423  ;       tick counters for all CPU's on the system. The monitored
                          0423  ;       system is a multiprocessing system, but the user requested
                          0423  ;       that the display and/or summary show combined values for
                          0423  ;       the processor modes (/NOCPU).
                          0423  ;
                          0423  ; CALLING SEQUENCE:
                          0423  ;
                          0423  ;       BSBW    COMBINE_MODES
                          0423  ;
                          0423  ; INPUTS:
                          0423  ;
                          0423  ;       None
                          0423  ;
                          0423  ; IMPLICIT INPUTS:
                          0423  ;
                          0423  ;       R6 = Pointer to MODES CDB
                          0423  ;       R8 = Pointer to CURRENT collection buffer
                          0423  ;
                          0423  ; OUTPUTS:
                          0423  ;
                          0423  ;       None
                          0423  ;
                          0423  ; IMPLICIT OUTPUTS:
                          0423  ;
                          0423  ;       The first 7 longwords in the data portion of the collection buffer
                          0423  ;       will contain combined mode counter values for all CPU's on the system.
                          0423  ;
                          0423  ; ROUTINE VALUE:
                          0423  ;
                          0423  ;       None
                          0423  ;
                          0423  ; SIDE EFFECTS:
                          0423  ;
                          0423  ;       Registers R0, R1, R2 destroyed.
                          0423  ;
                          0423  ;--
                          0423
                          0423
                          0423  COMBINE_MODES:
                          0423
     50   18 A6     D0    0423          MOVL    CDB$L_ECOUNT(R6),R0     ; Get number of modes for display
 51   58    0D      C1    0427          ADDL3   #MNR_CLS$K_HSIZE,R8,R1  ; Compute start addr of 1st set of counters
    52    6140      DE    042B          MOVAL   (R1)[R0],R2             ; Compute start addr of 2nd set of counters
                          042F  10$:
        81   82     C0    042F          ADDL2   (R2)+,(R1)+             ; Combine 2nd set with 1st set
           FA 50    F5    0432          SOBGTR  R0,10$                  ; ... for all counters
                          0435
                    05    0435          RSB                             ; Return
```

```
                              0436         .SBTTL  QUAD_LT_QUAD - Compare Two Quadwords
                              0436
                              0436  ;++
                              0436  ;
                              0436  ; FUNCTIONAL DESCRIPTION:
                              0436  ;
                              0436  ;        This routine is called by PL/I routines to compare two unsigned
                              0436  ;        quadword values (such as system time values). The routine answers
                              0436  ;        the question: Is the first value less than the second value?
                              0436  ;        The value YES or NO is placed in R0 upon exit.
                              0436  ;
                              0436  ; CALLING SEQUENCE:
                              0436  ;
                              0436  ;        CALLS #2,QUAD_LT_QUAD
                              0436  ;
                              0436  ; INPUTS:
                              0436  ;
                              0436  ;        4(AP) - address of first quadword value
                              0436  ;
                              0436  ;        8(AP) - address of second quadword value
                              0436  ;
                              0436  ; IMPLICIT INPUTS:
                              0436  ;
                              0436  ;        None
                              0436  ;
                              0436  ; OUTPUTS:
                              0436  ;
                              0436  ;        Routine value below.
                              0436  ;
                              0436  ; IMPLICIT OUTPUTS:
                              0436  ;
                              0436  ;        None
                              0436  ;
                              0436  ; ROUTINE VALUE:
                              0436  ;
                              0436  ;        R0 = YES if first quadword value is less than the second quadword value.
                              0436  ;        R0 = NO  otherwise
                              0436  ;
                              0436  ; SIDE EFFECTS:
                              0436  ;
                              0436  ;        None
                              0436  ;
                              0436  ;--
                              0436
                              0436
                        0000  0436  .ENTRY  QUAD_LT_QUAD,   ^M<>
                              0438
          50    04 AC    7D   0438          MOVQ    4(AP),R0              ; Pointers in R0 and R1
       04 A1    04 A0    D1   043C           MPL    4(R0),4(R1)           ; First < Second (hi-order) ?
                   0C    1F   0441          BLSSU   20$                   ; Yes -- answer is YES
                   05    1A   0443          BGTRU   10$                   ; No -- answer is NO
             61    60    D1   0445          CMPL    (R0),(R1)             ; Check low order longword
                   05    1F   0448          BLSSU   20$                   ; Go answer YES
                              044A  10$:
             50    00    D0   044A          MOVL    #NO,R0                ; First not less than second
                   03    11   044D          BRB     QLQ_RET               ; Go return
                              044F  20$:
```

```
50    01    D0  044F   1794          MOVL    #YES,R0                 ; First less than second
                0452   1795 QLQ_RET:
            04  0452   1796          RET                             ; Return with value in R0
```

```
                              0453          .SBTTL  QUAD_EQ_0 - Compare Quadword = 0
                              0453
                              0453  ;++
                              0453  ;
                              0453  ; FUNCTIONAL DESCRIPTION:
                              0453  ;
                              0453  ;       This routine is called by PL/I routines to compare an unsigned
                              0453  ;       quadword values (such as a system time value) with the quadword
                              0453  ;       value 0. The routine answers the question: Is the quadword value
                              0453  ;       equal to 0?
                              0453  ;
                              0453  ;       The value YES or NO is placed in R0 upon exit.
                              0453  ;
                              0453  ; CALLING SEQUENCE:
                              0453  ;
                              0453  ;       CALLS #1,QUAD_EQ_0
                              0453  ;
                              0453  ; INPUTS:
                              0453  ;
                              0453  ;       4(AP) - address of quadword value
                              0453  ;
                              0453  ; IMPLICIT INPUTS:
                              0453  ;
                              0453  ;       None
                              0453  ;
                              0453  ; OUTPUTS:
                              0453  ;
                              0453  ;       Routine value below.
                              0453  ;
                              0453  ; IMPLICIT OUTPUTS:
                              0453  ;
                              0453  ;       None
                              0453  ;
                              0453  ; ROUTINE VALUE:
                              0453  ;
                              0453  ;       R0 = YES if quadword value is equal to 0.
                              0453  ;       R0 = NO  otherwise
                              0453  ;
                              0453  ; SIDE EFFECTS:
                              0453  ;
                              0453  ;       None
                              0453  ;
                              0453  ;--
                              0453
                              0453
                        0000  0453  .ENTRY  QUAD_EQ_0,      ^M<>
                              0455
      50   04 BC    7D  0455          MOVQ    @4(AP),R0               ; Quadword value in R0, R1
              50    D5  0459          TSTL    R0                      ; Right half = 0?
              09    12  045B          BNEQU   10$                     ; No -- answer NO
              51    D5  045D          TSTL    R1                      ; Yes -- left half = 0?
              05    12  045F          BNEQU   10$                     ; No -- answer NO
      50      01    D0  0461          MOVL    #YES,R0                 ; Answer YES
              03    11  0464          BRB     QEZ_RET                 ; Go exit
                        0466  10$:
      50      00    D0  0466          MOVL    #NO,R0                  ; Answer NO
                        0469  QEZ_RET:
```

```
 04  0469  1855          RET                                     ; Return with value in R0
```

```
                                046A       .SBTTL  MPCHECK - Check system for MP capability
                                046A
                                046A  ;++
                                046A  ;
                                046A  ; FUNCTIONAL DESCRIPTION:
                                046A  ;
                                046A  ;        This routine is called by the EXECUTE_REQUEST PL/I routine
                                046A  ;        to determine whether or not the running system has MP
                                046A  ;        (multiprocessing) capability.
                                046A  ;
                                046A  ;        The value YES or NO is placed in R0 upon exit.
                                046A  ;
                                046A  ; CALLING SEQUENCE:
                                046A  ;
                                046A  ;        CALLS #0,QUAD_EQ_0
                                046A  ;
                                046A  ; INPUTS:
                                046A  ;
                                046A  ;        None
                                046A  ;
                                046A  ; IMPLICIT INPUTS:
                                046A  ;
                                046A  ;        EXE$GB_CPUTYPE -- CPU type. Assume type 1 = 780.
                                046A  ;
                                046A  ;        EXE$GL_RPB -- Address of Restart Parameter Block.
                                046A  ;
                                046A  ; OUTPUTS:
                                046A  ;
                                046A  ;        Routine value below.
                                046A  ;
                                046A  ; IMPLICIT OUTPUTS:
                                046A  ;
                                046A  ;        None
                                046A  ;
                                046A  ; ROUTINE VALUE:
                                046A  ;
                                046A  ;        R0 = YES if the system has MP capability
                                046A  ;        R0 = NO  otherwise
                                046A  ;
                                046A  ; SIDE EFFECTS:
                                046A  ;
                                046A  ;        None
                                046A  ;
                                046A  ;--
                                046A
                                046A
                          0000  046A  .ENTRY  MPCHECK,        ^M<>
                                046C
01    00000000'GF           91  046C          CMPB    G^EXE$GB_CPUTYPE,#1       ; 780 processor?
                  11        12  0473          BNEQ    10$                       ; No -- go answer NO
50    00000000'GF           D0  0475          MOVL    G^EXE$GL_RPB,R0           ; Get Restart Parameter Block ptr
   05 30 A0       0B        E1  047C          BBC     #RPB$V_MPM,RPB$L_BOOTR5(R0),10$ ; Br if no multi-port mem
            50    01        D0  0481          MOVL    #YES,R0                   ; Answer YES
                  03        11  0484          BRB     20$                       ; Go exit
                                0486  10$:
            50    00        D0  0486          MOVL    #NO,R0                    ; Answer NO
                                0489  20$:
```

```
 04  0489  1914         RET                               ; Return with value in R0
```

```
                              048A          .SBTTL  WRITE_PROC_RECORDS - Write PROCESSES class records
                              048A
                              048A   ;++
                              048A   ;
                              048A   ; FUNCTIONAL DESCRIPTION:
                              048A   ;
                              048A   ;       This routine is called to write out a group of class
                              048A   ;       records containing the data in a PROCESSES collection
                              048A   ;       buffer.
                              048A   ;
                              048A   ; CALLING SEQUENCE:
                              048A   ;
                              048A   ;       CALLS #2,WRITE_PROCS_RECCRDS
                              048A   ;
                              048A   ; INPUTS:
                              048A   ;
                              048A   ;       4(AP) - address of collection buffer
                              048A   ;
                              048A   ;       8(AP) - longword size of a data block (for a single process)
                              048A   ;
                              048A   ; IMPLICIT INPUTS:
                              048A   ;
                              048A   ;       None
                              048A   ;
                              048A   ; OUTPUTS:
                              048A   ;
                              048A   ;       None
                              048A   ;
                              048A   ; IMPLICIT OUTPUTS:
                              048A   ;
                              048A   ;       As many class records as are required are written to the
                              048A   ;       recording file for this PROCESSES collection buffer.
                              048A   ;
                              048A   ; ROUTINE VALUE:
                              048A   ;
                              048A   ;       R0 = NORMAL, or error status, if any
                              048A   ;
                              048A   ; SIDE EFFECTS:
                              048A   ;
                              048A   ;       None
                              048A   ;
                              048A   ;--
                              048A
                              048A
                        0FFC  048A   .ENTRY  WRITE_PROC_RECORDS,     ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              048C
     57    04 AC    D0        048C          MOVL    4(AP),R7                        ; Get pointer to collection buffer
     58    11 A7    D0        0490          MOVL    <MNR_CLS$K_HSIZE+MNR_PRO$L_PCTINT>(R7),R8
                              0494                                                  ; Get num of processes in coll buffer
   58   0016'CF     D1        0494          CMPL    W^PROCS_PER_REC,R8              ; Need more than 1 class record?
            25      19        0499          BLSS    10$                             ; Yes -- go break it up
                              049B          ALLOC   8,R1,R2                         ; No -- alloc a descriptor on stack
  62    08 AC    58    C5     04A8          MULL3   R8,8(AP),(R2)                   ; Calculate size of data area
        62       15    C0     04AD          ADDL2   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE>,(R2)
                              04B0                                                  ; Add in class hdr and PROCESSES prefix
     04 A2       57    D0     04B0          MOVL    R7,4(R2)                        ; Load addr of coll buff into descr
                 52    DD     04B4          PUSHL   R2                              ; Push descriptor address
     00000000'EF   01   FB  04B6  1973          CALLS   #1,WRITE_RECORD            ; ... and write the record
              00C6   31  04BD  1974          BRW     WPR_RET                    ; Go return with status of WRITE_RECORD
```

```
                                04C0      ;
                                04C0      ; Break up collection buffer into a group of records and write
                                04C0      ; each into the recording file. All except the last will have the
                                04C0      ; MNR_CLS$V_CONT flag set indicating that data for this interval
                                04C0      ; continues in the next record.
                                04C0      ;
                                04C0      ;
                                04C0      ; Register Usage:
                                04C0      ;
                                04C0      ;       R6 =  # of full records to write
                                04C0      ;       R7 =  collection buffer index pointer
                                04C0      ;       R8 =  # of processes in the collection buffer;
                                04C0      ;               also, # of processes in the final record
                                04C0      ;       R9 =  pointer to write buffer data area
                                04C0      ;       R10 = size of data portion of write buffer
                                04C0      ;       R11 = pointer to write buffer descriptor
                                04C0      ;
                                04C0
                                04C0      10$:
     56   58   0016'CF     C7   04C0                  DIVL3   W^PROCS_PER_REC,R8,R6    ; Get number of full records to write
     51   56   0016'CF     C5   04C6                  MULL3   W^PROCS_PER_REC,R6,R1    ; Calculate # of procs in ...
               58   51     C2   04CC                  SUBL2   R1,R8                    ; ... final record
 5A   08 AC    0016'CF     C5   04CF                  MULL3   W^PROCS_PER_REC,8(AP),R10 ; Get size of data portion of write buff
      51   5A       15     C1   04D6                  ADDL3   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE>,R10,R1
                                04DA                                                   ; Compute write buffer size
                                04DA                  ALLOC   8,R0,R11                 ; Get a write buffer descr on stack
               6B   51     D0   04E7                  MOVL    R1,(R11)                 ; Move in write buff size
                 04 AB     DF   04EA                  PUSHAL  4(R11)                   ; Push addr of write buffer ptr
                    6B     DF   04ED                  PUSHAL  (R11)                    · Push addr of write buffer size
       00000000'GF  02     FB   04EF                  CALLS   #2,G^LIB$GET_VM          ; Get the write buffer
                 03 50     E8   04F6                  BLBS    R0,20$                   ; Continue if status OK
                  008A     31   04F9                  BRW     WPR_RET                  ; Return with status if failed
                                04FC      20$:
         001A'CF    6B     7D   04FC                  MOVQ    (R11),W^PROC_WRI_BUFD    ; Save descriptor for later cleanup
            59   04 AB     D0   0501                  MOVL    4(R11),R9                ; Get ptr to write buffer
      69   67       15     28   0505                  MOVC3   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE>,(R7),(R9)
                                0509                                                   ; Move class hdr & prefix to write buff
               57   15     C0   0509                  ADDL2   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE>,R7 ; Update coll buff ptr
        00 01  A9   00     E2   050C                  BBSS    #MNR_CLS$V_CONT,MNR_CLS$W_FLAGS(R9),30$ ; Set "continued" bit
               59   0D     C0   0511      30$:        ADDL2   #MNR_CLS$K_HSIZE,R9      ; Point to PROCESSES prefix
      69   0016'CF         D0   0514                  MOVL    W^PROCS_PER_REC,MNR_PRO$L_PCTREC(R9) ; Load # of procs this rec
               59   08     C0   0519                  ADDL2   #MNR_PRO$K_PSIZE,R9      ; Point to data portion of write buffer
```

```
                             051C      ;
                             051C      ; Loop until all processes which fit into full records
                             051C      ; have been handled. On each time through the loop, move
                             051C      ; PROCS_PER_REC processes from the collection buffer to
                             051C      ; the write buffer and write it out. The class header
                             051C      ; portion and the PROCESSES prefix portion of the write
                             051C      ; buffer will be identical for all these records.
                             051C      ;
                             051C      ;
                             051C      40$:
      69   67   5A     28    051C              MOVC3   R10,(R7),(R9)              ; Move group of procs from coll to write
           57   5A     C0    0520              ADDL2   R10,R7                     ; Update coll buff ptr
                5B     DD    0523              PUSHL   R11                        ; Stack ptr to write buffer descr
 00000000'EF    01     FB    0525              CALLS   #1,WRITE_RECORD            ; Write this record
           57 50       E9    052C              BLBC    R0,WPR_RET                 ; Go exit with status if failed
           EA 56       F5    052F              SOBGTR  R6,40$                     ; Loop back to do next group
                             0532      ;
                             0532      ; Build and write a final record for the "leftover"
                             0532      ; processes which didn't fit into one of the full records.
                             0532      ;
                             0532
       51    04 AB     D0    0532              MOVL    4(R11),R1                  ; Get pointer to write buff class hdr
    00 01 A1    00     E5    0536              BBCC    #MNR_CLS$V_CONT,MNR_CLS$W_FLAGS(R1),50$ ; Clear "continued" bit
           51   0D     C0    053B      50$:    ADDL2   #MNR_CLS$K_HSIZE,R1        ; Point to PROCESSES prefix
           61   58     D0    053E              MOVL    R8,MNR_PRO$L_PCTREC(R1)    ; Load # of procs in this record
   5A    58    08 AC   C5    0541              MULL3   8(AP),R8,R10               ; Compute size of data portion
                5A     D5    0544              TSTL    R10                        ; Any "leftover" data to write?
                04     13    0548              BEQL    60$                        ; No -- skip around the MOVC3
      69   67   5A     28    054A              MOVC3   R10,(R7),(R9)              ; Move leftovers from coll to write buff
                             054E      60$:
                             054E              ALLOC   8,R0,R1                    ; Allocate a descr for the write
    04 A1    04 AB     D0    055B              MOVL    4(R11),4(R1)               ; Get write buff ptr from previous descr
      61    5A   15    C1    0560              ADDL3   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE> R10,(R1)
                             0564                                                 ; Move write buff length into descr
                51     DD    0564              PUSHL   R1                         ; Stack descr ptr for write
 00000000'EF    01     FB    0566              CALLS   #1,WRITE_RECORD            ; Write the last record
           16 50       E9    056D              BLBC    R0,WPR_RET                 ; Leave with status if failed
                             0570      ;
                             0570      ; Free the virtual memory occupied by the write buffer
                             0570      ;
                             0570
             04 AB     DF    0570              PUSHAL  4(R11)                     ; Stack addr of write buffer ptr
                6B     DF    0573              PUSHAL  (R11)                      ; Stack addr of write buffer len
 00000000'GF    02     FB    0575              CALLS   #2,G^LIB$FREE_VM           ; Free the write buffer memory
           07 50       E9    057C              BLBC    R0,WPR_RET                 ; Leave with status if failed
 50    00000000'EF     D0    057F              MOVL    NORMAL,R0                  ; Indicate normal status
                             0586      WPR_RET:
                       04    0586              RET                                ; Return with status
```

```
                                 0587               .SBTTL  CVT_TO_DELTA - Convert Seconds to Delta
                                 0587
                                 0587     ;++
                                 0587     ;
                                 0587     ; FUNCTIONAL DESCRIPTION:
                                 0587     ;
                                 0587     ;         This routine is called to convert a positive seconds quantity
                                 0587     ;         supplied in a longword to a quadword delta time quantity.
                                 0587     ;
                                 0587     ; CALLING SEQUENCE:
                                 0587     ;
                                 0587     ;         CALLS #2,CVT_TO_DELTA
                                 0587     ;
                                 0587     ; INPUTS:
                                 0587     ;
                                 0587     ;         4(AP) - address of longword containing positive seconds quantity
                                 0587     ;
                                 0587     ;         8(AP) - address of quadword in which to store converted delta time.
                                 0587     ;
                                 0587     ; IMPLICIT INPUTS:
                                 0587     ;
                                 0587     ;         None
                                 0587     ;
                                 0587     ; OUTPUTS:
                                 0587     ;
                                 0587     ;         Quadword addressed by 8(AP) is loaded with converted delta time.
                                 0587     ;
                                 0587     ; IMPLICIT OUTPUTS:
                                 0587     ;
                                 0587     ;         None
                                 0587     ;
                                 0587     ; ROUTINE VALUE:
                                 0587     ;
                                 0587     ;         None
                                 0587     ;
                                 0587     ; SIDE EFFECTS:
                                 0587     ;
                                 0587     ;         None
                                 0587     ;
                                 0587     ;--
                                 0587
                                 0587
                           0000  0587     .ENTRY  CVT_TO_DELTA,   ^M<>
                                 0589
00   04 BC   FF676980 8F    7A   0589             EMUL    #-10*1000*1000,@4(AP),#0,@8(AP) ; That's all, folks
                   08 BC         0592
                             04  0594             RET
```

```
                              0595  2115          .SBTTL  COMPUTE_BOOTTIME - Compute System Time of Boot
                              0595  2116
                              0595  2117  ;++
                              0595  2118  ;
                              0595  2119  ; FUNCTIONAL DESCRIPTION:
                              0595  2120  ;
                              0595  2121  ;       This routine is called to compute the quadword time value
                              0595  2122  ;       representing the absolute time at which the monitored
                              0595  2123  ;       (running) system was booted. This is done by converting
                              0595  2124  ;       the EXE$GL_ABSTIM value (absolute number of seconds since
                              0595  2125  ;       boot) to a negative nuadword system time value and adding
                              0595  2126  ;       it to the current time (obtained via $GETTIM).
                              0595  2127  ;
                              0595  2128  ; CALLING SEQUENCE:
                              0595  2129  ;
                              0595  2130  ;       CALLS #0,COMPUTE_BOOTTIME
                              0595  2131  ;
                              0595  2132  ; INPUTS:
                              0595  2133  ;
                              0595  2134  ;       None
                              0595  2135  ;
                              0595  2136  ; IMPLICIT INPUTS:
                              0595  2137  ;
                              0595  2138  ;       EXE$GL_ABSTIM - longword containing positive number of seconds
                              0595  2139  ;                       since boot.
                              0595  2140  ;
                              0595  2141  ;       SPTR - pointer to SYI (System Information Area).
                              0595  2142  ;
                              0595  2143  ; OUTPUTS:
                              0595  2144  ;
                              0595  2145  ;       None
                              0595  2146  ;
                              0595  2147  ; IMPLICIT OUTPUTS:
                              0595  2148  ;
                              0595  2149  ;       MNR_SYI$Q_BOOTTIME loaded with boot time in system time units.
                              0595  2150  ;
                              0595  2151  ; ROUTINE VALUE:
                              0595  2152  ;
                              0595  21;3  ;       R0 = NORMAL, or failing system service status
                              0595  2154  ;
                              0595  2155  ; SIDE EFFECTS:
                              0595  2156  ;
                              0595  2157  ;       None
                              0595  2158  ;
                              0595  2159  ;--
                              0595  2160
                              0595  2161
                        0004  0595  2162  .E'ITRY  COMPUTE_BOOTTIME,       ^M<R2>
                              0597  2163
                              0597  2164          ALLOC   8,R0,R;                 ; Get quadword on stack
                              05A4  2165          $GETTIM_S TIMADR=(R2)           ; Put current time into it
                   26 50   E9 05AD  2166          BLBC    R0,10$                  ; Exit if error
00    00000000'EF  FF676980 8F   7A 05B0  2167    EMUL    #-10*1000*1000,EXE$GL_ABS IM,#0,R0
                         50       05BC
                              05BD  2168                                          ; Get delta quadword system time units
                              05BD  2169                                          ; ... since boot
              50    62     C0 05BD  2170          ADDL2   (R2),R0                 ; Add low-order current time
              51    04 A2   D8  05C0  2171          ADWC    4(R2),R1                        ; ... and high-order current time
        52  00000000'EF   D0  05C4  2172          MOVL    SPTR,R2                         ; Get pointer to SYI
              03 A2    50   7D  05CB  2173          MOVQ    R0,MNR_SYI$Q_BOOTTIME(R2)       ; Move in boot time
        50  00000000'EF   D0  05CF  2174          MOVL    NORMAL,R0                       ; Indicate success
                                05D6  2175 10$:
                            04  05D6  2176          RET                                     ; Return
```

```
                                05D7          .SBTTL  CLUS_NET_INFO - Get Cluster & Net Info
                                05D7
                                05D7    ;++
                                05D7    ;
                                05D7    ; FUNCTIONAL DESCRIPTION:
                                05D7    ;
                                05D7    ;       This routine is called to gather certain info about the
                                05D7    ;       cluster and/or network and place it into the System
                                05D7    ;       Information Area.
                                05D7    ;
                                05D7    ; CALLING SEQUENCE:
                                05D7    ;
                                05D7    ;       CALLS #0,CLUS_NET_INFO
                                05D7    ;
                                05D7    ; INPUTS:
                                05D7    ;
                                05D7    ;       None
                                05D7    ;
                                05D7    ; IMPLICIT INPUTS:
                                05D7    ;
                                05D7    ;       CLU$GL_CLUB -   If non-zero, then this node is a cluster member;
                                05D7    ;                       otherwise, it is not.
                                05D7    ;
                                05D7    ;       SPTR - pointer to SYI (System Information Area).
                                05D7    ;
                                05D7    ;       SCSNODE system parameter -- SCS node name in ASCII
                                05D7    ;       SYS$NODE logical name -- DECnet node name in ASCII
                                05D7    ;
                                05D7    ; OUTPUTS:
                                05D7    ;
                                05D7    ;       None
                                05D7    ;
                                05D7    ; IMPLICIT OUTPUTS:
                                05D7    ;
                                05D7    ;       SYI updated.
                                05D7    ;
                                05D7    ; ROUTINE VALUE:
                                05D7    ;
                                05D7    ;       R0 = NORMAL, or failing system service status
                                05D7    ;
                                05D7    ; SIDE EFFECTS:
                                05D7    ;
                                05D7    ;       None
                                05D7    ;
                                05D7    ;--
                                05D7
                                05D7
                          007C  05D7    .ENTRY  CLUS_NET_INFO,  ^M<R2,R3,R4,R5,R6>
                                05D9
     56   00000000'EF     D0    05D9            MOVL    SPTR,R6                 ; Get System Info Area pointer
                                05E0
                                05E0    ;
                                05E0    ; Set MNR_SYI$V_CLUSMEM if this node is a cluster member
                                05E0    ;
                                05E0
                                05E0            IFCLSTR 20$                     ; Br if a cluster member
      00 01 A6    00      E5    05E8            BBCC    #MNR_SYI$V_CLUSMEM, -   ; Else indicate not member
```

```
                           05ED                        MNR_SYI$W_FLAGS(R6),10$
                 05   11   05ED    10$:      BRB       30$                         ; ... and go continue
   00 01 A6      00   E2   05EF    20$:      BBSS      #MNR_SYI$V_CLUSMEM, -       ; Indicate a cluster member
                           05F4                        MNR_SYI$W_FLAGS(R6),30$
                           05F4    30$:
                           05F4
                           05F4    ;
                           05F4    ; Call $GETSYI to get CPU type and node name (from SCSNODE parameter)
                           05F4    ;
                           05F4
                           05F4              ALLOC     56,R0,R2                    ; Allocate local temp storage
                           0601
                           0601    ;
                           0601    ; Set up item descriptor for node name
                           0601    ;
                           0601
         62      10   B0   0601              MOVW      #16,(R2)                    ; Move in length of buffer
   02 A2       10D9 8F   B0   0604           MOVW      #SYI$_NODENAME,2(R2)        ; Move in item identifier
    04 A2      1C A2   DE   060A             MOVAL     28(R2),4(R2)                ; Move in ptr to buffer
    08 A2      2C A2   DE   060F             MOVAL     44(R2),8(R2)                ; Move in ptr to length word
                           0614
                           0614    ;
                           0614    ; Set up item descriptor for CPU type
                           0614    ;
                           0614
      0C A2      04   B0   0614              MOVW      #4,12(R2)                   ; Move in length of buffer
   0E A2       2000 8F   B0   0618           MOVW      #SYI$_CPU,14(R2)            ; Move in item identifier
    10 A2      30 A2   DE   061E             MOVAL     48(R2),16(R2)               ; Move in ptr to buffer
    14 A2      34 A2   DE   0623             MOVAL     52(R2),20(R2)               ; Move in ptr to length word
               18 A2   D4   0628             CLRL      24(R2)                      ; Indicate end of item list
                           062B              $GETSYIW_S ITMLST=(R2)                ; Get the CPU type & SCSNODF node name
               03 50   E8   063E             BLBS      R0,40$                      ; Continue if status OK
                0082   31   0641             BRW       CNI_RET                     ; Go exit if error
                           0644    40$:
    26 A6      30 A2   D0   0644             MOVL      48(R2),MNR_SYI$L_CPUTYPE(R6) ; Pick up CPU type
                           0649    ;
                           0649    ; Now process node name
                           0649    ;
                           0649
       53      04 A2   D0   0649             MOVL      4(R2),R3                    ; Set up ptr to 1st byte of string
       54      08 A2   D0   064D             MOVL      8(R2),R4                    ; Set up ptr to length word
                  64   B5   0651             TSTW      (R4)                        ; Check for null node name
                  43   12   0653             BNEQ      70$                         ; Br if not null
                           0655
                           0655    ;
                           0655    ; $GETSYI returned a null node name, so try to get the
                           0655    ; DECnet node name by translating logical name SYS$NODE.
                           0655    ;
                           0655
                           0655              ALLOC     15,R2,R3                    ; Get the result buffer on stack
                           0662              ALLOC     2,R0,R4                     ; ... and a word for actual length
                           066F
                           066F              $TRNLOG_S LOGNAM=W^SYSNOD_NAM, RSLBUF=(R2), -
                           066F                        RSLLEN=(R4), DSBMSK=#^B110
                           0684
                           0684                                                    ; Translate SYS$NODE in system table
               3F 50   E9   0684             BLBC      R0,CNI_RET                  ; Exit if error
```

```
                                     0687  2292
          50  00000629 8F     D1     0687  2293              CMPL    #SS$_NOTRAN,R0           ; SYS$NODE logical name found?
                         08   12     068E  2294              BNEQ    70$                      ; Br if yes
                      0E A6   7C     0690  2295              CLRQ    MNR_SYI$T_NODENAME(R6)   ; Otherwise, clear out node name
                      16 A6   7C     0693  2296              CLRQ    MNR_SYI$T_NODENAME+8(R6) ; .......
                         27   11     0696  2297              BRB     CNI_SUCC                 ; ... and go exit
                                     0698  2298
                                     0698  2299 ;
                                     0698  2300 ; A node name has been found. Now eliminate leading underscore and
                                     0698  2301 ; trailing double colons if present.
                                     0698  2302 ;
                                     0698  2303 ; At this point, R3 = addr of first character of node name, and
                                     0698  2304 ;                R4 = addr of a non-zero word length field
                                     0698  2305 ;
                                     0698  2306
                                     0698  2307 70$:
                                     0698  2308
              63    5F 8F     91     0698  2309              CMPB    #^A/_/,(R3)                    ; Is 1st character of string an _??
                         04   12     069C  2310              BNEQU   80$                            ; No -- go check for trailing colons
                         53   D6     069E  2311              INCL    R3                             ; Yes -- redefine string address ...
                         64   B7     06A0  2312              DECW    (R4)                           ; ... and length to eliminate _
                                     06A2  2313 80$:
                   55    64   3C     06A2  2314              MOVZWL  (R4),R5                        ; Get current length of string
              55   FE A345    9E     06A5  2315              MOVAB   -2(R3)[R5],R5                  ; Compute addr of end of string - 2
              65   3A3A 8F    B1     06AA  2316              CMPW    #^A/::/,(R5)                   ; Are last 2 characters colons?
                         03   12     06AF  2317              BNEQU   90$                            ; No -- go move string to SYI
                   64    02   A2     06B1  2318              SUBW2   #2,(R4)                        ; Yes -- shorten string by 2 bytes
                                     06B4  2319 90$:
                0E A6    64   90     06B4  2320              MOVB    (R4),MNR_SYI$T_NODENAME(R6); Get length of node name
  0F A6   0F   00   63   64   2C     06B8  2321              MOVC5   (R4),(R3),#0,#15, -
                                     06BF  2322                      MNR_SYI$T_NODENAME+1(R6) ; ... and actual name string
                                     06BF  2323
                                     06BF  2324 CNI_SUCC:
          50  00000000'EF     D0     06BF  2325              MOVL    NORMAL,R0                      ; Indicate success
                                     06C6  2326 CNI_RET:
                              04     06C6  2327              RET                                    ; Return
```

```
                                06C7          .SBTTL  ADV_HOM_ITEM - Advance to next display item for homog
                                06C7
                                06C7  ;++
                                06C7  ;
                                06C7  ; FUNCTIONAL DESCRIPTION:
                                06C7  ;
                                06C7  ;       This routine is called by PL/I routines to advance the
                                06C7  ;       current (homogeneous) class to the next item for display.
                                06C7  ;
                                06C7  ; CALLING SEQUENCE:
                                06C7  ;
                                06C7  ;       CALLS #1,ADV_HOM_ITEM
                                06C7  ;
                                06C7  ; INPUTS:
                                06C7  ;
                                06C7  ;       4(AP) - address of CDB pointer for current display class
                                06C7  ;
                                06C7  ; IMPLICIT INPUTS:
                                06C7  ;
                                06C7  ;       None
                                06C7  ;
                                06C7  ; OUTPUTS:
                                06C7  ;
                                06C7  ;       None
                                06C7  ;
                                06C7  ; IMPLICIT OUTPUTS:
                                06C7  ;
                                06C7  ;       CDX$B_IDISCONSEC and CDX$B_IDISINDEX fields updated to
                                06C7  ;       indicate the next item to be displayed.
                                06C7  ;
                                06C7  ; ROUTINE VALUE:
                                06C7  ;
                                06C7  ;       SS$_NORMAL
                                06C7  ;
                                06C7  ; SIDE EFFECTS:
                                06C7  ;
                                06C7  ;       None
                                06C7  ;
                                06C7  ;--
                                06C7
                                06C7
                          001C  06C7  .ENTRY  ADV_HOM_ITEM,   ^M<R2,R3,R4>
                                06C9
                                06C9  ;
                                06C9  ; Bump CDX$B_IDISCONSEC so that next requested item is chosen for display.
                                06C9  ;
                                06C9
      52     04 BC     D0       06C9          MOVL    @4(AP),R2                 ; Get CDB address
      52     32 A2     D0       06CD          MOVL    CDB$A_CDX(R2),R2          ; Get CDX address
             07 A2     96       06D1          INCB    CDX$B_IDISCONSEC(R2)      ; Point to next consecutive item
   06 A2     07 A2     91       06D4          CMPB    CDX$B_IDISCONSEC(R2), -   ; Past final item?
                                06D9                  CDX$B_IDISCT(R2)
                04     15       06D9          BLEQ    10$                       ; Br if not
   07 A2        01     90       06DB          MOVB    #1,CDX$B_IDISCONSEC(R2)   ; Restart consec no. at 1st item
                                06DF
                                06DF  ;
                                06DF  ; Now update CDX$B_IDISINDEX to be in sync with the new value
                          06DF  ; of CDX$B_IDISCONSEC.
                          06DF  ;
                          06DF
                          06DF  10$:
          51    10    D0  06DF         MOVL    #CDX$S_IBITS,R1           ; Init bit field size
                50    D4  06E2         CLRL    R0                        ; Init start position
                          06E4
       54    07 A2    9A  06E4         MOVZBL  CDX$B_IDISCONSEC(R2),R4 ; Get item display consec no.
                          06E8  40$:
53    62    51  50    EA  06E8         FFS     R0,R1,CDX$W_IBITS(R2),R3 ; Search for next item number
                          06ED                                       ; R3 contains item number if found
      50    53  01    C1  06ED         ADDL3   #1,R3,R0                  ; Compute next starting ...
      51    10  50    C3  06F1         SUBL3   R0,#CDX$S_IBITS,R1        ; ... position and field size
             F0 54    F5  06F5         SOBGTR  R4,40$                    ; Loop until item number is found
                          06F8
        08 A2   53    90  06F8         MOVB    R3,CDX$B_IDISINDEX(R2)    ; ... and store it away
                          06FC
          50    01    D0  06FC         MOVL    #SS$_NORMAL,R0            ; Normal status
                      04  06FF         RET                               ; Return
```

```
0700         .SBTTL  COLLECTION - Collect into CURRENT Buffer
0700
0700  ;++
0700  ;
0700  ; FUNCTIONAL DESCRIPTION:
0700  ;
0700  ;        This routine is called to collect data for the current
0700  ;        class into the buffer pointed to by the CURRENT register
0700  ;        (R8). The data is obtained either from the running system
0700  ;        (live collection) or from an input recording file (playback).
0700  ;
0700  ; CALLING SEQUENCE:
0700  ;
0700  ;        BSBW    COLLECTION
0700  ;
0700  ; INPUTS:
0700  ;
0700  ;        None
0700  ;
0700  ; IMPLICIT INPUTS:
0700  ;
0700  ;        4(AP) - address of byte containing class number
0700  ;
0700  ;        Registers:
0700  ;
0700  ;                R6  = CDB pointer
0700  ;                R7  = MRB pointer
0700  ;                R8  = CURRENT collection buffer pointer
0700  ;                R11 = MCA pointer
0700  ;
0700  ;                If non-STD class, R8 has to have been set up before
0700  ;                entry to this routine.
0700  ;
0700  ; OUTPUTS:
0700  ;
0700  ;        None
0700  ;
0700  ; IMPLICIT OUTPUTS:
0700  ;
0700  ;        CURRENT buffer filled with data for this class.
0700  ;
0700  ;        On first collection event, MRB$Q_BEGINNING is loaded with
0700  ;          the system time of the first collection.
0700  ;
0700  ; ROUTINE VALUE:
0700  ;
0700  ;        R0 = NORMAL, or error status, if any
0700  ;
0700  ; SIDE EFFECTS:
0700  ;
0700  ;        Registers R0,R1,R2,R3,R4,R5 destroyed.
0700  ;
0700  ;--
```

```
                             0700  COLLECTION:
                             0700
           37 43 A7   03  E1 0700          BBC     #MRB$V_PLAYBACK,MRB$W_FLAGS(R7),30$ ; If not collecting from
                             0705                                                  ; ... a file, go do live collection
                             0705
           08 4B A6   04  E0 0705          BBS     #CDB$V_STD,CDB$L_FLAGS(R6),10$ ; Branch if standard class
                             070A  ;
                             070A  ; PLAYBACK -- Non-standard Class (PROCESSES)
                             070A  ;
                 0080     30 070A          BSBW    COLL_NONSTD              ; Fill coll buff from playback file
                 7C 50    E9 070D          BLBC    R0,COLL_RSB              ; Premature EOF error possible here
                    52    11 0710          BRB     COLL_COMM                ; Go join common collection code
                             0712
                             0712  ;
                             0712  ; PLAYBACK -- Standard Class
                             0712  ;
                             0712  ; Pick up (with MOVC5) the collection buffer previously placed in a
                             0712  ; fixed location by the PL/I read routine.
                             0712  ;
                             0712
                             0712  10$:
              0600 8F     BB 0712          PUSHR   #^M<R9,R10>              ; Save regs 9 and 10
           59   04 AB     D0 0716          MOVL    MCA$A_INPUT_PTR(R11),R9  ; Get base of read buffer
                             071A
                             071A  ;
                             071A  ; Compute size of collection buffer
                             071A  ;
                             071A
           5A   20 A6     3C 071A          MOVZWL  CDB$W_BLKLEN(R6),R10     ; Get length of a data block
        0A 4B A6      05  E1 071E          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),20$ ; Br if hetero
      5A   000000C8 8F    C4 0723          MULL2   #MAXELTS,R10             ; One data block for each element
              5A   08     C0 072A          ADDL2   #MNR_HOM$K_PSIZE,R10     ; Chip in with prefix size
                             072D  20$:
              5A   0D     C0 072D          ADDL2   #MNR_CLS$K_HSIZE,R10     ; Add in header size
                             0730
    68    5A    00   69  6B   2C 0730          MOVC5   MCA$L_INPUT_LEN(R11),(R9),#0,R10,(R8) ; Move to CURRENT buffer
              0600 8F     BA 0736          POPR    #^M<R9,R10>              ; Restore regs 9 and 10
                    28    11 073A          BRB     COLL_COMM                ; Go join common collection code
                             073C
                             073C  ;
                             073C  ; LIVE COLLECTION
                             073C  ;
                             073C  ; Create argument list (on stack) for $CMKRNL call to FETCH.
                             073C  ;
                             073C
                             073C  30$:
           7E   58   0D   C1 073C          ADDL3   #MNR_CLS$K_HSIZE,R8,-(SP) ; Stack addr of beg of variable
                             0740                                       ; ... portion of collection buffer
                    56    DD 0740          PUSHL   R6                       ; Stack CDB address
                    02    DD 0742          PUSHL   #2                       ; Top off list with arg count
              54    5E    D0 0744          MOVL    SP,R4                    ; Remember arg list addr for $CMKRNL
                             0747  ;
                             0747  ; Fill collection buffer header
                             0747  ;
           68   04 BC     90 0747          MOVB    @4(AP),MNR_CLS$B_TYPE(R8) ; Collect class number into buffer
                01 A8     B4 074B          CLRW    MNR_CLS$W_FLAGS(R8)      ; Zero out flags ...
                08 A8     B4 074E          CLRW    MNR_CLS$W_RESERVED(R8)   ; ... and reserved word
```

```
                              0751                 $CMKRNL_S  FETCH,(R4)            ; Call FETCH to fill in rest of buffer
                              075E
             5E    0C     C0  075E                 ADDL    #12,SP                   ; Restore stack
                28 50     E9  0761                 BLBC    R0,COLL_PSB              ; NOPRIV or other error possible here
                              0764
                              0764   COLL_COMM:                                     ; Common code for playback or live collectio
   50   00000000'EF       D0  0764                 MOVL    NORMAL,R0                ; Indicate successful status
      0B A8     34 AB     B0  076B                 MOVW    MCA$L_CONSEC_REC(R11),MNR_CLS$W_RESERVED(R8)
                              0770                                                  ; Store (low-order word of)
                              0770                                                  ; ... consec no in coll buff
                0C AB     D5  0770                 TSTL    MCA$L_COLLCNT(R11)       ; First collection?
                   17     12  0773                 BNEQ    COLL_RSB                 ; No -- just return
      30 AB     04 BC     91  0775                 CMPB    @4(AP),MCA$B_FIRSTC(R11) ; First class?
                   10     12  077A                 BNEQU   COLL_RSB                 ; No -- just return
         67     03 A8     7D  077C                 MOVQ    MNR_CLS$Q_STAMP(R8),MRB$Q_BEGINNING(R7)
                              0780                                                  ; Yes -- save start time of collection
      07 43 A7     01     E1  0780                 BBC     #MRB$V_RECORD,MRB$W_FLAGS(R7),COLL_RSB ; Exit if not recording
   00000000'EF     00     FB  0785                 CALLS   #0,WRITE_HEADER          ; Write recording file header
                              078C
                              078C   COLL_RSB:
                          05  078C                 RSB                              ; Return
```

```
                                078D  ;
                                078D  ; Since the PROCESSES class can have class data which spans several
                                078D  ; records, loop reading all PROCESSES records for this interval,
                                078D  ; concatenating the data portions of the records into the collection
                                078D  ; buffer.
                                078D  ;
                                078D
                                078D  COLL_NONSTD:
                                078D
            0780 8F       BB    078D          PUSHR   #^M<R7,R8,R9,R10>          ; Save registers
         59   04 AB       D0    0791          MOVL    MCA$A_INPUT_PTR(R11),R9 ; Get base of read buffer
      68   69   6B        28    0795          MOVC3   MCA$L_INPUT_LEN(R11),(R9),(R8) ; 1st class rec to CURRENT buffer
    3F 01 A9    00        E1    0799          BBC     #MNR_CLS$V_CONT,MNR_CLS$W_FLAGS(R9),30$ ; Done if only 1 class rec
           58   6B        C0    079E          ADDL2   MCA$L_INPUT_LEN(R11),R8 ; Point to end of coll buffer
      5A   59   15        C1    07A1          ADDL3   #<MNR_CLS$K_HSIZE+MNR_PRO$K_PSIZE>,R9,R10
                                07A5                                             ; Point to PROCESSES data in read buff
                57        D4    07A5          CLRL    R7                         ; Ensure high-order word is clear ...
                                07A7                                             ; ... for later use
                                07A7  ;
                                07A7  ; Read the next PROCESSES class record for this interval
                                07A7  ;
                                07A7
                                07A7  10$:
  7E    00000000'8F       D0    07A7          MOVL    #SKIP_TO_CLASS,-(SP)       ; Indicate next class rec is desired
                5E        DD    07AE          PUSHL   SP                         ; Stack indicator for call
  00000000'EF   01        FB    07B0          CALLS   #1,READ_INPUT              ; Read next class record
           5E   04        C0    07B7          ADDL2   #4,SP                      ; Pop stack
     09 32 AB   03        E1    07BA          BBC     #MCA$V_EOF,MCA$W_FLAGS(R11),20$ ; Continue if not end-of-file
  50    00000000'8F       D0    07BF          MOVL    #MNR$_PREMEOF,R0           ; Else indicate error ...
                1C        11    07C6          BRB     40$                        ; ... and go return
                                07C8  20$:
      50   59   0D        C1    07C8          ADDL3   #MNR_CLS$K_HSIZE,R9,R0     ; Point to PROCESSES prefix in read buff
   57   20 A6   60        A5    07CC          MULW3   MNR_PRO$L_PCTREC(R0),CDB$W_BLKLEN(R6),R7
                                07D1                                             ; Calc size of next move
      68   6A   57        28    07D1          MOVC3   R7,(R10),(R8)              ; Next class rec to CURRENT buffer
           58   57        C0    07D5          ADDL2   R7,R8                      ; Point to end of coll buffer
    CA 01 A9    00        E0    07D8          BBS     #MNR_CLS$V_CONT,MNR_CLS$W_FLAGS(R9),10$ ; If cont, go read next
                                07DD  30$:
  50    00000000'EF       D0    07DD          MOVL    NORMAL,R0                  ; Indicate normal status
                                07E4  40$:
            0780 8F       BA    07E4          POPR    #^M<R7,R8,R9,R10>          ; Restore registers
                          05    07E8          RSB                                ; Return
                                07E9
```

```
                    07E9          .SBTTL  FILL_HETERO_STATS  -  Fill the STATS Buffer
                    07E9
                    07E9  ;++
                    07E9  ;
                    07E9  ; FUNCTIONAL DESCRIPTION:
                    07E9  ;
                    07E9  ;       This routine is called to fill in the STATS buffer
                    07E9  ;       for the current (heterogeneous) class, given the current
                    07E9  ;       and previous buffers. For COUNT-type data items, each
                    07E9  ;       STATS item is computed as CURRENT minus PREVIOUS; for
                    07E9  ;       LEVEL-type data items, each STATS item is merely copied
                    07E9  ;       from CURRENT. This process may be viewed as "levelizing"
                    07E9  ;       the counts.
                    07E9  ;
                    07E9  ; CALLING SEQUENCE:
                    07E9  ;
                    07E9  ;       BSBW FILL_HETERO_STATS
                    07E9  ;
                    07E9  ; INPUTS:
                    07E9  ;
                    07E9  ;       R6 =    address of CDB (Class Descriptor Block)
                    07E9  ;
                    07E9  ;       R8 =    address of CURRENT buffer
                    07E9  ;
                    07E9  ;       R9 =    address of PREVIOUS buffer
                    07E9  ;
                    07E9  ;       R10 =   address of buffer block
                    07E9  ;
                    07E9  ; IMPLICIT INPUTS:
                    07E9  ;
                    07E9  ;       PERFTABLE - table describing each data item, indexed by
                    07E9  ;                       item number ( * entry size)
                    07E9  ;
                    07E9  ; OUTPUTS:
                    07E9  ;
                    07E9  ;       None
                    07E9  ;
                    07E9  ; IMPLICIT OUTPUTS:
                    07E9  ;
                    07E9  ;       STATS buffer is filled.
                    07E9  ;
                    07E9  ; ROUTINE VALUE:
                    07E9  ;
                    07E9  ;       None
                    07E9  ;
                    07E9  ; SIDE EFFECTS:
                    07E9  ;
                    07E9  ;       Registers R0,R1,R2,R3,R4,R5 destroyed.
                    07E9  ;
                    07E9  ;--
                    07E9
                    07E9
                    07E9  FILL_HETERO_STATS:
                    07E9
      0800 8F   BB  07E9          PUSHR   #^M<R11>                ; Save reg 11
 50    18 A6    D0  07ED          MOVL    CDB$L_ECOUNT(R6),R0     ; Get number of items to fetch
 51    1C A6    D0  07F1          MOVL    CDB$A_ITMSTR(R6),R1     ; Address of item-number string
```

```
      52   59   0D   C1  07F5  2640           ADDL3   #MNR_CLS$K_HSIZE,R9,R2  ; Calc start of items for PREVIOUS
      53   58   0D   C1  07F9  2641           ADDL3   #MNR_CLS$K_HSIZE,R8,R3  ; Calc start of items for CURRENT
       54    08 AA   D0  07FD  2642           MOVL    MBP$A_STATS(R10),R4     ; Get pointer to STATS
                5B   D4  0801  2643           CLRL    R11                     ; Clear loop counter
                         0803  2644 10$:
           55   81   9A  0803  2645           MOVZBL  (R1)+,R5                ; Get next item number
           55   11   C4  0806  2646           MULL    #IDB$K_ILENGTH,R5       ; Compute index into IDB table
  55     0000'CF45   9E  0809  2647           MOVAB   W^PERFTABLE[R5],R5      ; Address of IDB for this item
                         080F  2648           CASE    IDB$W_ISIZE(R5),<20$,30$,40$>,W ; Select on proper size
                         081A  2649
                         081A  2650 20$:
0000'8F     0A A5    B1  081A  2651           CMPW    IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
                0D   12  0820  2652           BNEQ    25$                     ; No -- assume level type
      64   83   82   83  0822  2653           SUBB3   (R2)+,(R3)+,(R4)        ; Compute byte diff into STATS buff
                02   18  0826  2654           BGEQ    23$                     ; Br if difference OK
                64   94  0828  2655           CLRB    (R4)                    ; Counter has decreased; use 0 value
                         082A  2656 23$:
           84   64   9A  082A  2657           MOVZBL  (R4),(R4)+              ; Zero-extend to longword
                45   11  082D  2658           BRB     50$
                         082F  2659 25$:
           84   83   9A  082F  2660           MOVZBL  (R3)+,(R4)+             ; Move CURRENT byte level to STATS
                82   95  0832  2661           TSTB    (R2)+                   ; Auto-increment PREVIOUS buffer
                3E   11  0834  2662           BRB     50$
                         0836  2663 30$:
0000'8F     0A A5    B1  0836  2664           CMPW    IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
                0D   12  083C  2665           BNEQ    35$                     ; No -- assume level type
      64   83   82   A3  083E  2666           SUBW3   (R2)+,(R3)+,(R4)        ; Compute word diff into STATS buff
                02   18  0842  2667           BGEQ    33$                     ; Br if difference OK
                64   B4  0844  2668           CLRW    (R4)                    ; Counter has decreased; use 0 value
                         0846  2669 33$:
           84   64   3C  0846  2670           MOVZWL  (R4),(R4)+              ; Zero-extend to longword
                29   11  0849  2671           BRB     50$
                         084B  2672 35$:
           84   83   3C  084B  2673           MOVZWL  (R3)+,(R4)+             ; Move CURRENT word level to STATS
                82   B5  084E  2674           TSTW    (R2)+                   ; Auto-increment PREVIOUS buffer
                22   11  0850  2675           BRB     50$
                         0852  2676 40$:
0000'8F     0A A5    B1  0852  2677           CMPW    IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
                0B   12  0858  2678           BNEQ    45$                     ; No -- assume level type
      84   83   82   C3  085A  2679           SUBL3   (R2)+,(R3)+,(R4)+       ; Compute long diff into STATS buff
                14   18  085E  2680           BGEQ    50$                     ; Br if difference OK
             FC A4   D4  0860  2681           CLRL    -4(R4)                  ; Counter has decreased; use 0 value
                0F   11  0863  2682           BRB     50$
                         0865  2683 45$:
             10 A5   95  0865  2684           TSTB    IDB$B_FLAGS(R5)         ;Is this a computed item?
                05   13  0868  2685           BEQL    47$                     ;branch if not
              0010   30  086A  2686           BSBW    GET_COMPUTED_ITEMS      ;otherwise need to do some special calc.
                05   11  086D  2687           BRB     50$                     ;continue
                         086F  2688 47$:
           84   83   D0  086F  2689           MOVL    (R3)+,(R4)+             ; Move CURRENT longword level to STATS
                82   D5  0872  2690           TSTL    (R2)+                   ; Auto-increment PREVIOUS buffer
                         0874  2691 50$:
        8B 5B   50   F2  0874  2692           AOBLSS  R0,R11,10$              ; Loop until done
           0800 8F   BA  0878  2693           POPR    #^M<R11>                ; Restore reg 11
                     05  087C  2694           RSB                             ; ... and return
                         087D  2695
                         087D  2696
```

```
    087D  2697
```

```
                    087D  2699 ;++
                    087D  2700 ; GET_COMPUTED_ITEMS
                    087D  2701 ;
                    087D  2702 ;
                    087D  2703 ;      Computed items are items which are not recorded, but are transformations
                    087D  2704 ;      of other items.  This routine makes the appropriate transformations and
                    087D  2705 ;      loads the STATS buffer for computed items.
                    087D  2706 ;
                    087D  2707 ;              R2 = address of current longword in PREVIOUS buffer
                    087D  2708 ;              R3 = address of current longword in CURRENT buffer
                    087D  2709 ;              R4 = address of current longword in STAT buffer
                    087D  2710 ;
                    087D  2711
                    087D  2712 ; Currently, the only type of computed item supported by Monitor is
                    087D  2713 ; the percentage item.  Here are the steps involved in adding a new percentage
                    087D  2714 ; item:
                    087D  2715 ;                  -  The item is flagged as a percentage by including
                    087D  2716 ;                     FLAGS=IDB$M_PCNT in the BLDIDB for the item.  The item
                    087D  2717 ;                     Type should be LEVEL.
                    087D  2718 ;                  -  The next two items following the percentage in the class
                    087D  2719 ;                     will be used to calculate the percentage value.  The formula
                    087D  2720 ;                     is
                    087D  2721 ;                          Percentage value = (Item1 * 100)/Item2
                    087D  2722 ;
                    087D  2723 ;                     For example, in the case of XQPCACHE percentages,
                    087D  2724 ;                     Item1 = Cache hits and item2 = (Cache hits + Cache misses)
                    087D  2725 ;                     The items used to calculate the percentage must be longwords.
                    087D  2726 ;
                    087D  2727 ;                  -  The percentage and item2 will be displayed; item1 will
                    087D  2728 ;                     not be displayed.
                    087D  2729 ;
                    087D  2730 ; Computed items currently may only be included in Standard Heterogeneous
                    087D  2731 ; classes.
                    087D  2732
                    087D  2733 ; To add a new type of computed item, define a new bitmask for IDB$B_FLAGS
                    087D  2734 ; (for-instance IDB$M_RATIO) and add the necessary code to do the new
                    087D  2735 ; computation.  GET_COMPUTED_ITEMS will be called by FILL_HETERO_STATS anytime
                    087D  2736 ; the IDB$B_FLAGS field is nonzero, so this routine is one place you will want
                    087D  2737 ; to add new code.  Depending on the nature of the item, you may need to add
                    087D  2738 ; code elsewhere.  (for-instance, TEMPLATE, to prevent display of items used
                    087D  2739 ; in the calculation, and INTAVE to do special processing to obtain the
                    087D  2740 ; average for the item).
                    087D  2741
                    087D  2742 ; Computed items are included in the item count for a class, but are not
                    087D  2743 ; included in the CDB$W_BLKLEN value.
                    087D  2744 ;--
                    087D  2745
                    087D  2746 GET_COMPUTED_ITEMS:
                    087D  2747 ;
                    087D  2748 ; If this is a percent, compute the long dif for the following two items
                    087D  2749 ; and do the calculation to end up with a level in the stats buff.
                    087D  2750 ; Also, increment the stats buff pointer but not current or prev.
                    087D  2751 ;
     00C0 8F   BB   087D  2752          PUSHR   #^M<R6,R7>                ;save some registers so we can use them
1D 10 A5   00   E1   0881  2753          BBC     #IDB$V_PCNT, IDB$B_FLAGS(R5), 20$      ;branch if not a percent
           64   D4   0886  2754          CLRL    (R4)              ; zero longword for this pcnt item in STAT buff
56   63    62   C3   0888  2755          SUBL3   (R2),(R3),R6      ; Current minus previous for item1 into R6
```

```
         57   04 A3   04 A2   C3  088C  2756          SUBL3   4(R2),4(R3),R7  ; Current minus previous for item2 into R7
                         57   D5  0892  2757          TSTL    R7              ; Is the sum nonzero?
                         0B   13  0894  2758          BEQL    10$             ; branch if it is zero to skip percent calc.
                                  0896  2759 ;
                                  0896  2760 ; now create a percentage value: (item1 * 100)/(item1 + item2)
                                  0896  2761 ;
           56   00000064 8F   C4  0896  2762          MULL2   #100,R6         ;
              64   56    57   C7  089D  2763          DIVL3   R7,R6,(R4)      ;move the percentage result into the STAT buff.
                                  08A1  2764 10$:
                         84   D5  08A1  2765          TSTL    (R4)+           ;auto-increment the STAT buffer pointer.
                                  08A3  2766 20$:
                    00C0 8F   BA  08A3  2767          POPR    #^M<R6,R7>      ;restore the registers
                              05  08A7  2768          RSB                     ;return
                                  08A8  2769
```

```
                           08A8        .SBTTL  FILL_PCSTATS_BUFF - Fill PCSTATS Buffer from STATS Buffer
                           08A8
                           08A8  ;++
                           08A8  ;
                           08A8  ; FUNCTIONAL DESCRIPTION:
                           08A8  ;
                           08A8  ;       Fill the PCSTATS buffer with integer values representing
                           08A8  ;       tenths of percent for each item in the STATS buffer.
                           08A8  ;
                           08A8  ; INPUTS:
                           08A8  ;
                           08A8  ;       R6  - CDB pointer
                           08A8  ;       R10 - Buffer block pointer
                           08A8  ;
                           08A8  ; IMPLICIT INPUTS:
                           08A8  ;
                           08A8  ;       STATS buffer containing levels to be "percentized."
                           08A8  ;
                           08A8  ; OUTPUTS:
                           08A8  ;
                           08A8  ;       None
                           08A8  ;
                           08A8  ; IMPLICIT OUTPUTS:
                           08A8  ;
                           08A8  ;       PCSTATS buffer filled with integer values representing
                           08A8  ;       tenths of percent.
                           08A8  ;
                           08A8  ; ROUTINE VALUE:
                           08A8  ;
                           08A8  ;       None
                           08A8  ;
                           08A8  ; SIDE EFFECTS:
                           08A8  ;
                           08A8  ;       Registers R0, R1, R3, R4 destroyed.
                           08A8  ;--
                           08A8
                           08A8  FILL_PCSTATS_BUFF:
                           08A8
              53   18 AA   D0  08A8        MOVL    MBP$A_PCSTATS(R10),R3    ; Load PCSTATS buffer pointer
              54   08 AA   D0  08AC        MOVL    MBP$A_STATS(R10),R4      ; Load STATS buffer pointer
              50   18 A6   D0  08B0        MOVL    CDB$L_ECOUNT(R6),R0      ; Get number of elements
                      51   D4  08B4        CLRL    R1                       ; Clear accumulator
                               08B6  10$:
                   51   84 C0  08B6        ADDL2   (R4)+,R1                 ; Add next item
                   FA 50   F5  08B9        SOBGTR  R0,10$                   ; Continue until STATS summed
              54   08 AA   D0  08BC        MOVL    MBP$A_STATS(R10),R4      ; Re-load STATS buffer pointer
              50   18 A6   D0  08C0        MOVL    CDB$L_ECOUNT(R6),R0      ; ... and number of elements
                      51   D5  08C4        TSTL    R1                       ; Zero sum?
                      08   12  08C6        BNEQ    30$                      ; No -- go calc percentages
                               08C8  20$:
                   83   84 D0  08C8        MOVL    (R4)+,(R3)+              ; Yes -- simply move zeroes
                   FA 50   F5  08CB        SOBGTR  R0,20$                   ; ... into PCSTATS buffer
                      0E   11  08CE        BRB     40$                      ; ... and go exit
                               08D0  30$:
  63    84  000003E8 8F    C5  08D0        MULL3   #1000,(R4)+,(R3)         ; Multiply value by 1000 ... and
                   83   51 C6  08D8        DIVL2   R1,(R3)+                 ; Divide by sum, leaving tenths of %
                   F2 50   F5  08DB        SOBGTR  R0,30$                   ; Continue for all items
```

```
    08DE  2828 40$:
05  08DE  2829          RSB                                 ; Return
```

```
                    08DF          .SBTTL  COMPUTE_STATS - Statistical Computations on STATS
                    08DF
                    08DF    ;++
                    08DF    ;
                    08DF    ; FUNCTIONAL DESCRIPTION:
                    08DF    ;
                    08DF    ;       Replace each count item in the STATS buffer with a computed
                    08DF    ;       (floating-point) rate per second. Also, replace each item in
                    08DF    ;       the MIN and MAX buffers with the corresponding item from the
                    08DF    ;       STATS buffer if it establishes a new minimum or maximum value.
                    08DF    ;
                    08DF    ; INPUTS:
                    08DF    ;
                    08DF    ;       R6  - CDB pointer
                    08DF    ;       R7  - MRB pointer
                    08DF    ;       R8  - CURRENT buffer pointer
                    08DF    ;       R9  - PREVIOUS buffer pointer
                    08DF    ;       R10 - Buffer block pointer
                    08DF    ;       R11 - MCA pointer
                    08DF    ;
                    08DF    ; IMPLICIT INPUTS:
                    08DF    ;
                    08DF    ;       STATS buffer
                    08DF    ;
                    08DF    ;       HOMOG_TYPE - word containing item type code of current
                    08DF    ;                    item for homogeneous class.
                    08DF    ;
                    08DF    ;       PERFTABLE - table describing each data item, indexed by
                    08DF    ;                       item number ( * entry size).
                    08DF    ;
                    08DF    ;       MCA$L_INTTICKS - clock ticks during interval just finished
                    08DF    ;
                    08DF    ; OUTPUTS:
                    08DF    ;
                    08DF    ;       None
                    08DF    ;
                    08DF    ; IMPLICIT OUTPUTS:
                    08DF    ;
                    08DF    ;       Each count element in STATS buffer converted to floating rate/second.
                    08DF    ;
                    08DF    ;       Elements in MIN and MAX buffers updated if new min and max
                    08DF    ;       values were achieved in the interval just completed.
                    08DF    ;
                    08DF    ; ROUTINE VALUE:
                    08DF    ;
                    08DF    ;       None
                    08DF    ;
                    08DF    ; SIDE EFFECTS:
                    08DF    ;
                    08DF    ;       Registers R0,R1,R2,R3,R4,R5 destroyed.
                    08DF    ;--
                    08DF
                    08DF    COMPUTE_STATS:
                    08DF
0300 8F    BB       08DF          PUSHR   #^M<R8,R9>              ; Save regs
                    08E3    ;
                    08E3    ; Load registers for upcoming buffer manipulations
```

```
                             08E3  2888 ;
                             08E3  2889
                             08E3  2890 10$:
         52    08 AA    D0   08E3  2891          MOVL     MBP$A_STATS(R10),R2      ; Load addr of first STATS item
         58    0C AA    D0   08E7  2892          MOVL     MBP$A_MIN(R10),R8        ; Load addr of first MIN item
         59    10 AA    D0   08EB  2893          MOVL     MBP$A_MAX(R10),R9        ; Load addr of first MAX item
         50    1C A6    D0   08EF  2894          MOVL     CDB$A_ITMSTR(R6),R0      ; Load addr of item-number string
         53    18 A6    D0   08F3  2895          MOVL     CDB$L_ECOUNT(R6),R3      ; Load number of items in STATS
                  51    D4   08F7  2896          CLRL     R1                       ; Clear loop counter
                             08F9  2897 20$:
      0B 4B A6    05    E1   08F9  2898          BBC      #CDB$V_HOMOG,CDB$L_FLAGS(R6),25$  ; Br if heterogeneous
  0000'8F    00E2'CF    B1   08FE  2899          CMPW     W^HOMOG_TYPE,#COUNT_TYPE ; Is this homog item a count?
                  1B    12   0905  2900          BNEQU    30$                      ; No -- assume level
                  33    11   0907  2901          BRB      50$                      ; Yes -- go process count
                             0909  2902 25$:
      14 4B A6    00    E1   0909  2903          BBC      #CDB$V_CTPRES,CDB$L_FLAGS(R6),30$ ; Skip type check if no counts
            55    80    9A   090E  2904          MOVZBL   (R0)+,R5                 ; Get next item number
            55    11    C4   0911  2905          MULL     #IDB$K_ILENGTH,R5        ; Compute index into IDB table
      55   0000'CF45    9E   0914  2906          MOVAB    W^PERFTABLE[R5],R5       ; Address of IDB for this item
   0000'8F     0A A5    B1   091A  2907          CMPW     IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
                  1A    13   0920  2908          BEQLU    50$                      ; Yes -- go compute rate
                             0922  2909 ;
                             0922  2910 ; Update MIN and MAX buffers for this item (level).
                             0922  2911 ;
                             0922  2912
                             0922  2913 30$:
       6841     6241    D1   0922  2914          CMPL     (R2)[R1],(R8)[R1]        ; Check minimum
                  05    18   0927  2915          BGEQ     40$                      ; Branch if not less
       6841     6241    D0   0929  2916          MOVL     (R2)[R1],(R8)[R1]        ; Else insert new minimum
                             092E  2917 40$:
       6941     6241    D1   092E  2918          CMPL     (R2)[R1],(R9)[R1]        ; Check maximum
                  33    15   0933  2919          BLEQ     70$                      ; Branch if not more
       6941     6241    D0   0935  2920          MOVL     (R2)[R1],(R9)[R1]        ; Else insert new maximum
                  2C    11   093A  2921          BRB      70$                      ; ... and go loop
                             093C  2922 ;
                             093C  2923 ; Compute rate/second for this count item, replacing count in
                             093C  2924 ; STATS buffer.
                             093C  2925 ;
                             093C  2926
                             093C  2927 50$:
         54     6241    4E   093C  2928          CVTLF    (R2)[R1],R4              ; Get floating value over interval
         55    08 AB    4E   0940  2929          CVTLF    MCA$L_INTTICKS(R11),R5   ; Get floating ticks over interval
   55  000043C8 8F      46   0944  2930          DIVF2    #100,R5                  ; Get floating seconds over interval
     6241   54    55    47   094B  2931          DIVF3    R5,R4,(R2)[R1]           ; Floating rate/second into STATS
                             0950  2932 ;
                             0950  2933 ; Update MIN and MAX buffers for this item (count).
                             0950  2934 ;
                             0950  2935
       6841     6241    51   0950  2936          CMPF     (R2)[R1],(R8)[R1]        ; Check minimum
                  05    18   0955  2937          BGEQ     60$                      ; Branch if not less
       6841     6241    50   0957  2938          MOVF     (R2)[R1],(R8)[R1]        ; Else insert new minimum
                             095C  2939 60$:
       6941     6241    51   095C  2940          CMPF     (R2)[R1],(R9)[R1]        ; Check maximum
                  05    15   0961  2941          BLEQ     70$                      ; Branch if not more
       6941     6241    50   0963  2942          MOVF     (R2)[R1],(R9)[R1]        ; Else insert new maximum
                             0968  2943 70$:
         8D 51    53    F2   0968  2944          AOBLSS   R3,R1,20$                ; Loop for each item in STATS
```

```
                     096C   2945
0300 8F      BA      096C   2946             POPR     #^M<R8,R9>                 ; Restore regs
             05      0970   2947             RSB                                 ; Return
```

```
                          0971  2949          .SBTTL  UPD_PC_MIN_MAX - Update Percent Min/Max Buffers
                          0971  2950
                          0971  2951  ;++
                          0971  2952  ;
                          0971  2953  ; FUNCTIONAL DESCRIPTION:
                          0971  2954  ;
                          0971  2955  ;       Replace each item in the PCMIN and PCMAX buffers with the
                          0971  2956  ;       corresponding item from the PCSTATS buffer if it establishes
                          0971  2957  ;       a new minimum or maximum value.
                          0971  2958  ;
                          0971  2959  ; CALLING SEQUENCE:
                          0971  2960  ;
                          0971  2961  ;       BSBW UPD_PC_MIN_MAX
                          0971  2962  ;
                          0971  2963  ; INPUTS:
                          0971  2964  ;
                          0971  2965  ;       R6  - CDB pointer
                          0971  2966  ;       R10 - Buffer block pointer
                          0971  2967  ;       R11 - MCA pointer
                          0971  2968  ;
                          0971  2969  ; IMPLICIT INPUTS:
                          0971  2970  ;
                          0971  2971  ;       PCSTATS buffer containing percent values derived from
                          0971  2972  ;       most recently collected levels.
                          0971  2973  ;
                          0971  2974  ; OUTPUTS:
                          0971  2975  ;
                          0971  2976  ;       None
                          0971  2977  ;
                          0971  2978  ; IMPLICIT OUTPUTS:
                          0971  2979  ;
                          0971  2980  ;       Items in PCMIN and PCMAX buffers updated according y.
                          0971  2981  ;
                          0971  2982  ; ROUTINE VALUE:
                          0971  2983  ;
                          0971  2984  ;       None
                          0971  2985  ;
                          0971  2986  ; SIDE EFFECTS:
                          0971  2987  ;
                          0971  2988  ;       Registers R0,R1,R2,R3,R4,R5 destroyed.
                          0971  2989  ;--
                          0971  2990
                          0971  2991
                          0971  2992  UPD_PC_MIN_MAX:
                          0971  2993
     53    18 AA    D0    0971  2994          MOVL    MBP$A_PCSTATS(R10),R3   ; Load PCSTATS ptr
     54    1C AA    D0    0975  2995          MOVL    MBP$A_PCMIN(R10),R4     ; Load PCMIN ptr
     55    20 AA    D0    0979  2996          MOVL    MBP$A_PCMAX(R10),R5     ; Load PCMAX ptr
     50    18 A6    D0    097D  2997          MOVL    CDB$L_ECOUNT(R6),R0     ; Get element count
              52    D4    0981  2998          CLRL    R2                      ; Clear loop counter
```

```
                          0983  3000 ;
                          0983  3001 ; Replace minimum and maximum (if necessary) for each item
                          0983  3002 ;
                          0983  3003
                          0983  3004 10$:
 6442    6342     D1      0983  3005          CMPL     (R3)[R2],(R4)[R2]        ; Check minimum
           05     18      0988  3006          BGEQ     20$                      ; Branch if not less
 6442    6342     D0      098A  3007          MOVL     (R3)[R2],(R4)[R2]        ; Else insert new minimum
                          098F  3008 20$:
 6542    6342     D1      098F  3009          CMPL     (R3)[R2],(R5)[R2]        ; Check maximum
           05     15      0994  3010          BLEQ     30$                      ; Branch if not more
 6542    6342     D0      0996  3011          MOVL     (R3)[R2],(R5)[R2]        ; Else insert new maximum
                          099B  3012 30$:
  E4 52    50     F2      099B  3013          AOBLSS   R0,R2,10$                ; Loop for each item in PCSTATS
                          099F  3014
                  05      099F  3015          RSB                               ; Return
```

```
09A0          .SBTTL  DISPLAY_INIT - Init for Display Output
09A0
09A0   ;++
09A0   ;
09A0   ; FUNCTIONAL DESCRIPTION:
09A0   ;
09A0   ;       This routine performs initialization for DISPLAY output.
09A0   ;
09A0   ; CALLING SEQUENCE:
09A0   ;
09A0   ;       CALLS #0, DISPLAY_INIT
09A0   ;
09A0   ; INPUTS:
09A0   ;
09A0   ;       None.
09A0   ;
09A0   ; IMPLICIT INPUTS:
09A0   ;
09A0   ;       MRBPTR - pointer to MRB (Monitor Request Block)
09A0   ;       MCAPTR - pointer to MCA (Monitor Communication Area)
09A0   ;
09A0   ; OUTPUTS:
09A0   ;
09A0   ;       BARCHAR loaded with appropriate bar character.
09A0   ;
09A0   ;       SYSOUT_TYPE loaded with display output device class.
09A0   ;
09A0   ;               The four classes are:
09A0   ;
09A0   ;                       DEC_CRT
09A0   ;                       VT5X
09A0   ;                       OTHER_VID
09A0   ;                       HARDCOPY (including disk file)
09A0   ;
09A0   ;       MCA$V_VIDEO set if display device is a video terminal.
09A0   ;
09A0   ;       MCA$V_GRAPHICS set if display device is a VT55.
09A0   ;
09A0   ;       ATTRIBMSK (DEC_CRT video attribute mask) cleared.
09A0   ;
09A0   ; IMPLICIT OUTPUTS:
09A0   ;
09A0   ;       None.
09A0   ;
09A0   ; ROUTINE VALUE:
09A0   ;
09A0   ;       R0 = NORMAL or failing status from SCRPKG routine.
09A0   ;
09A0   ; SIDE EFFECTS:
09A0   ;
09A0   ;       None
09A0   ;
09A0   ; REGISTER USAGE:
09A0   ;
09A0   ;       R7   = MRB pointer
09A0   ;       R11  = MCA pointer
09A0   ;
```

```
                              09A0   ;--
                              09A0
                              09A0
                       088C   09A0   .ENTRY  DISPLAY_INIT,    ^M<R2,R3,R7,R11>
                              09A2
              272B'CF   D4    09A2          CLRL     W^ATTRIBMSK               ; Turn off DEC_CRT attributes
     57   00000000'EF   D0    09A6          MOVL     MRBPTR,R7                 ; Load MRB pointer
     5B   00000000'EF   D0    09AD          MOVL     MCAPTR,R11                ; Load MCA pointer
       00 43 A7    08   E2    09B4          BBSS     #MRB$V_DIS_CL_REQ,MRB$W_FLAGS(R7),5$
                              09B9   5$:                                       ; Indicate display cleanup required
                 00EC   30    09B9          BSBW     COMMON_INIT               ; Do initialization common with SUMMARY
                              09BC
                              09BC   ;
                              09BC   ; Establish SCRPKG output stream
                              09BC   ;
                              09BC
                20 A7   D5    09BC          TSTL     MRB$A_DISPLAY(R7)         ; Is there a display file-spec?
                   12   13    09BF          BEQL     10$                       ; No -- continue
                20 A7   DD    09C1          PUSHL    MRB$A_DISPLAY(R7)         ; Yes -- stack it for SET_OUTPUT
                   01   DD    09C4          PUSHL    #1                        ; ... along with a stream identifier
     00000000'GF   02   FB    09C6          CALLS    #2,G^SCR$SET_OUTPUT       ; Establish output stream
                03 50   E8    09CD          BLBS     R0,10$                    ; Continue if status OK
                 0087   31    09D0          BRW      110$                      ; Go exit if SCR$SET_OUTPUT failed
                              09D3
                              09D3   ;
                              09D3   ; Establish bar character, sysout device type.
                              09D3   ;
                              09D3   ; Set MCA$V_VIDEO bit if a video terminal.
                              09D3   ;
                              09D3   ; Set MVA$V_GRAPHICS if a VT55 terminal.
                              09D3   ;
                              09D3
                              09D3   10$:
         0005'CF   2A   90    09D3          MOVB     #DEF_BAR,W^BARCHAR        ; Start out with default bar char
         2615'CF   03   90    09D8          MOVB     #OTHER_VID,W^SYSOUT_TYPE  ; ... and assume "other video" sysout type
                              09DD          ALLOC    9,R1,R2                   ; Allocate 9 bytes for screen info
                   52   DD    09EA          PUSHL    R2                        ; Stack addr of screen info buffer
     00000000'GF   01   FB    09EC          CALLS    #1,G^SCR$SCREEN_INFO      ; Get screen info (dev type)
                03 50   E8    09F3          BLBS     R0,20$                    ; Continue if status OK
                 0061   31    09F6          BRW      110$                      ; Go exit if SCR$SCREEN_INFO failed
                              09F9   20$:
        53    08 A2     90    09F9          MOVB     SCR$B_DEVTYPE(R2),R3      ; Save SYS$OUTPUT device type
        07 62      00   E1    09FD          BBC      #SCR$V_SCREEN,SCR$L_FLAGS(R2),30$ ; If not video, keep going
     09 32 AB      04   E2    0A01          BBSS     #MCA$V_VIDEO,MCA$W_FLAGS(R11),40$ ; Otherwise, indicate video
                   07   11    0A06          BRB      40$                       ; ... and continue
                              0A08   30$:
         2615'CF   02   90    0A08          MOVB     #HARDCOPY,W^SYSOUT_TYPE   ; Set hardcopy type
                   3B   11    0A0D          BRB      100$                      ; ... and go take def bar char
                              0A0F   40$:
        53    41 8F     91    0A0F          CMPB     #DT$_VT55,R3              ; Is it a VT55 ?
                   0C   12    0A13          BNEQU    60$                       ; No -- go check for other types
     00 32 AB      05   E2    0A15          BBSS     #MCA$V_GRAPHICS,MCA$W_FLAGS(R11),50$
                              0A1A                                             ; Yes -- indicate VT55-style graphics
                              0A1A   50$:
         2615'CF   01   90    0A1A          MOVB     #VT5X,W^SYSOUT_TYPE       ; Indicate VT5x series
                   23   11    0A1F          BRB      90$                       ; Go set special bar char
        07 62      06   E1    0A21   60$:   BBC      #SCR$V_DECCRT,SCR$L_FLAGS(R2),70$ , If not DEC CRT, keep going
```

```
      2615'CF    00   90  0A25  3131          MOVB    #DEC_CRT,W^SYSOUT_TYPE    ; Set DEC CRT
                 18   11  0A2A  3132          BRB     90$                       ; ... and go set special bar char
       53   60   8F   91  0A2C  3133 70$:     CMPB    #DT$_VT100,R3             ; Is it a VT100 ?
                 07   12  0A30  3134          BNEQU   80$                       ; No -- more checking
      2615'CF    00   90  0A32  3135          MOVB    #DEC_CRT,W^SYSOUT_TYPE    ; Yes -- set DEC CRT
                 0B   11  0A37  3136          BRB     90$                       ; ... and go set special bar char
                          0A39  3137 80$:
       53   40   8F   91  0A39  3138          CMPB    #DT$_VT52,R3              ; Is it a VT52 ?
                 0B   12  0A3D  3139          BNEQU   100$                      ; No -- take def bar char and type
      2615'CF    01   90  0A3F  3140          MOVB    #VT5X,W^SYSOUT_TYPE       ; Yes -- indicate VT5x series
  0005'CF   61   8F   90  0A44  3141 90$:     MOVB    #VID_BAR,W^BARCHAR        ; Indicate special bar char
                          0A4A  3142
                          0A4A  3143 100$:
            009A      30  0A4A  3144          BSBW    MOVE_BARS                 ; Move bar char into several places
                          0A4D  3145
                          0A4D  3146
                          0A4D  3147 ;
                          0A4D  3148 ; Kick off buffering mode for the Screen Package
                          0A4D  3149 ;
                          0A4D  3150
       000020AF'EF    7F  0A4D  3151          PUSHAQ  SCRDSC                    ; Push this routine's buffer addr
    00000000'GF  01   FB  0A53  3152          CALLS   #1,G^LIB$SET_BUFFER       ; Set buffering mode
                          0A5A  3153 110$:
                      04  0A5A  3154          RET                               ; Return with R0 = status
```

```
                          0A5B          .SBTTL  SUMMARY_INIT - Init for Summary Output
                          0A5B
                          0A5B  ;++
                          0A5B  ;
                          0A5B  ; FUNCTIONAL DESCRIPTION:
                          0A5B  ;
                          0A5B  ;       This routine performs initialization for SUMMARY output.
                          0A5B  ;
                          0A5B  ; CALLING SEQUENCE:
                          0A5B  ;
                          0A5B  ;       CALLS #0, SUMMARY_INIT
                          0A5B  ;
                          0A5B  ; INPUTS:
                          0A5B  ;
                          0A5B  ;       None.
                          0A5B  ;
                          0A5B  ; IMPLICIT INPUTS:
                          0A5B  ;
                          0A5B  ;       MRBPTR - pointer to MRB (Monitor Request Block)
                          0A5B  ;
                          0A5B  ; OUTPUTS:
                          0A5B  ;
                          0A5B  ;       BARCHAR loaded with appropriate bar character.
                          0A5B  ;
                          0A5B  ;       SYSOUT_TYPE loaded with display output device type.
                          0A5B  ;
                          0A5B  ;       ATTRIBMSK (DEC_CRT video attribute mask) cleared.
                          0A5B  ;
                          0A5B  ; IMPLICIT OUTPUTS:
                          0A5B  ;
                          0A5B  ;       None.
                          0A5B  ;
                          0A5B  ; ROUTINE VALUE:
                          0A5B  ;
                          0A5B  ;       R0 = NORMAL or failing status from SCRPKG routine.
                          0A5B  ;
                          0A5B  ; SIDE EFFECTS:
                          0A5B  ;
                          0A5B  ;       None
                          0A5B  ;
                          0A5B  ; REGISTER USAGE:
                          0A5B  ;
                          0A5B  ;       R7   = MRB pointer
                          0A5B  ;
                          0A5B  ;--
                          0A5B
                          0A5B
                    0080  0A5B  .ENTRY  SUMMARY_INIT,   ^M<R7>
                          0A5D
      272B'CF        D4   0A5D          CLRL    W^ATTRIBMSK                     ; Turn off DEC_CRT attributes
57  00000000'EF      D0   0A61          MOVL    MRBPTR,R7                       ; Load MRB pointer
  00 43 A7     09    E2   0A68          BBSS    #MRB$V_SUM_CL_REQ,MRB$W_FLAGS(R7), 5$
                          0A6D  5$:                                             ; Indicate summary cleanup required
  03 43 A7     00    E0   0A6D          BBS     #MRB$V_DISPLAY,MRB$W_FLAGS(R7),10$ ; Skip init if already done
             0033    30   0A72          BSBW    COMMON_INIT                     ; Do initialization common with DISPLAY
                          0A75
                          0A75  ;
```

```
                                0A75  ; Establish SCRPKG output stream
                                0A75  ;
                                0A75
                                0A75  10$:
              28 A7      DD     0A75          PUSHL   MRB$A_SUMMARY(R7)        ; Stack SUMMARY filespec for SET_OUTPUT
                 01      DD     0A78          PUSHL   #1                       ; ... along with a stream identifier
      00000000'GF   02   FB     0A7A          CALLS   #2,G^SCR$SET_OUTPUT      ; Establish output stream
              03 50      E8     0A81          BLBS    R0,20$                   ; Continue if status OK
               0020      31     0A84          BRW     40$                      ; Go exit if SCR$SET_OUTPUT failed
                                0A87
                                0A87  ;
                                0A87  ; Establish bar character, sysout device type.
                                0A87  ;
                                0A87
                                0A87  20$:
         0005'CF    2A   90     0A87          MOVB    #DEF_BAR,W^BARCHAR       ; Use default bar character
                    59   10     0A8C          BSBB    MOVE_BARS                ; Move bar char into several places
      00002615'EF   02   90     0A8E          MOVB    #HARDCOPY,SYSOUT_TYPE    ; Treat SYS$OUTPUT dev type as hardcopy
         00 43 A7   05   E2     0A95          BBSS    #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R7),30$
                                0A9A                                           ; Indicate output to file
                                0A9A  30$:
                                0A9A
                                0A9A  ;
                                0A9A  ; Kick off buffering mode for the Screen Package
                                0A9A  ;
                                0A9A
        000020AF'EF     7F      0A9A          PUSHAQ  SCRDSC                   ; Push this routine's buffer addr
      00000000'GF   01   FB     0AA0          CALLS   #1,G^LIB$SET_BUFFER      ; Set buffering mode
                                0AA7  40$:
                         04     0AA7          RET                              ; Return with R0 = status
```

```
                                  0AA8  3244 COMMON_INIT:
                                  0AA8  3245
                                  0AA8  3246 ;
                                  0AA8  3247 ; Do initialization for DISPLAY and SUMMARY options.
                                  0AA8  3248 ;
                                  0AA8  3249
         008D'CF    01      D0    0AA8  3250         MOVL    #SS$_NORMAL,W^PTS_STAT  ; Start off request with clean status
                                  0AAD  3251
                                  0AAD  3252 ;
                                  0AAD  3253 ; Set up footing display line with appropriate words
                                  0AAD  3254 ;
                                  0AAD  3255
    23F7'CF    23F5'CF      DE    0AAD  3256         MOVAL   W^BLANK_STR,W^FOOTP     ; Indicate blank string for PLAYBACK
       07 43 A7     03      E1    0AB4  3257         BBC     #MRB$V_PLAYBACK,MRB$W_FLAGS(R7),10$ ; Continue if no PLAYBACK
    23F7'CF    23C4'CF      DE    0AB9  3258         MOVAL   W^PLAY_STR,W^FOOTP      ; Indicate PLAYBACK string in footing
                                  0AC0  3259 10$:
    23FB'CF    23F5'CF      DE    0AC0  3260         MOVAL   W^BLANK_STR,W^FOOTS     ; Indicate blank string for SUMMARY
       07 43 A7     02      E1    0AC7  3261         BBC     #MRB$V_SUMMARY,MRB$W_FLAGS(R7),20$ ; Continue if no SUMMARY
    23FB'CF    23D3'CF      DE    0ACC  3262         MOVAL   W^SUMM_STR,W^FOOTS      ; Indicate SUMMARY string in footing
                                  0AD3  3263 20$:
    23FF'CF    23F5'CF      DE    0AD3  3264         MOVAL   W^BLANK_STR,W^FOOTR     ; Indicate blank string for RECORD
       07 43 A7     01      E1    0ADA  3265         BBC     #MRB$V_RECORD,MRB$W_FLAGS(R7),30$ ; Continue if no RECORD
    23FF'CF    23E5'CF      DE    0ADF  3266         MOVAL   W^REC_STR,W^FOOTR       ; Indicate RECORD string in footing
                                  0AE6  3267 30$:
                            05    0AE6  3268         RSB                             ; Return
                                  0AE7  3269
                                  0AE7  3270
                                  0AE7  3271 MOVE_BARS:                              ; Move bar char into several places
    26EA'CF    0005'CF      90    0AE7  3272         MOVB    W^BARCHAR,W^TOPBAR      ; Move bar char into TOP display line
 00000000'EF   0005'CF      90    0AEE  3273         MOVB    W^BARCHAR,BAR1          ; ... and into SYSTEM class FAO str
 00000000'EF   0005'CF      90    0AF7  3274         MOVB    W^BARCHAR,BAR2          ; .....
 00000000'EF   0005'CF      90    0B00  3275         MOVB    W^BARCHAR,BAR3          ; .....
 00000000'EF   0005'CF      90    0B09  3276         MOVB    W^BARCHAR,BAR4          ; .....
 00000000'EF   0005'CF      90    0B12  3277         MOVB    W^BARCHAR,BAR5          ; .....
 00000000'EF   0005'CF      90    0B1B  3278         MOVB    W^BARCHAR,BAR6          ; .....
 00000000'EF   0005'CF      90    0B24  3279         MOVB    W^BARCHAR,BAR7          ; .....
 00000000'EF   0005'CF      90    0B2D  3280         MOVB    W^BARCHAR,BAR8          ; .....
 00000000'EF   0005'CF      90    0B36  3281         MOVB    W^BARCHAR,BAR9          ; .....
 00000000'EF   0005'CF      90    0B3F  3282         MOVB    W^BARCHAR,BAR10         ; .....
 00000000'EF   0005'CF      90    0B48  3283         MOVB    W^BARCHAR,BAR11         ; .....
                                  0B51  3284
00000085'EF  00000000'EF    DE    0B51  3285         MOVAL   SYS_BOX_STR_G,SYS_BOX_STR_ADDR ; Choose graphic box str (SYSTEM)
00000083'EF  00000000'EF    B0    0B5C  3286         MOVW    SYS_BOX_STR_LEN_G,SYS_BOX_STR_LEN ; ... and its length
         2A  00000005'EF    91    0B67  3287         CMPB    BARCHAR,#DEF_BAR        ; This terminal use the def bar char?
                      16    12    0B6E  3288         BNEQ    10$                     ; No, all done
00000085'EF  00000000'EF    DE    0B70  3289         MOVAL   SYS_BOX_STR_H,SYS_BOX_STR_ADDR ; Choose hardcopy box str
00000083'EF  00000000'EF    B0    0B7B  3290         MOVW    SYS_BOX_STR_LEN_H,SYS_BOX_STR_LEN ; ... and its length
                                  0B86  3291 10$:
                            05    0B86  3292         RSB
```

```
0B87         .SBTTL  FILL_DISP_BUFF - Fill Display Buffer
0B87
0B87  ;++
0B87  ;
0B87  ; FUNCTIONAL DESCRIPTION:
0B87  ;
0B87  ;       This routine is called to fill the display buffer with values
0B87  ;       to be presented to FAOL for display of the screen for the current
0B87  ;       class. The address of the CDB for the current class is passed
0B87  ;       as the first parameter to this routine. The second parameter is
0B87  ;       the address of a quadword into which this routine will store the
0B87  ;       time stamp from the most recent collection buffer.
0B87  ;
0B87  ; CALLING SEQUENCE:
0B87  ;
0B87  ;       CALLS #2,FILL_DISP_BUFF
0B87  ;
0B87  ; INPUTS:
0B87  ;
0B87  ;        4(AP) - address of a pointer to the CDB (Class Descriptor Block)
0B87  ;                for the class to display.
0B87  ;
0B87  ;        8(AP) - address of quadword in which to store the time
0B87  ;                stamp from the most recent collection buffer.
0B87  ;
0B87  ; IMPLICIT INPUTS:
0B87  ;
0B87  ;       MCAPTR - pointer to MCA (Monitor Communication Area)
0B87  ;
0B87  ;       MRBPTR - pointer to MRB (Monitor Request Block)
0B87  ;
0B87  ;       PERFTABLE - table describing each data item, indexed by
0B87  ;                       item number ( * entry size)
0B87  ;
0B87  ;       FAOSTK - buffer into which to store values for later FAOL call.
0B87  ;
0B87  ; OUTPUTS:
0B87  ;
0B87  ;       Quadword pointed to by 8(AP) is filled with time stamp from
0B87  ;       most recent collection buffer.
0B87  ;
0B87  ; IMPLICIT uUTPUTS:
0B87  ;
0B87  ;       Display buffer (FAOSTK) buffer is filled.
0B87  ;
0B87  ;       For the non-standard class (PROCESSES),
0B87  ;       MCA$L_PROC_DISP is filled with the count
0B87  ;       of processes to display.
0B87  ;
0B87  ; ROUTINE VALUE:
0B87  ;
0B87  ;       R0 = SS$_NORMAL
0B87  ;
0B87  ; SIDE EFFECTS:
0B87  ;
0B87  ;       None
0B87  ;
```

```
                           0B87  ; REGISTER USAGE:
                           0B87  ;
                           0B87  ;         R5 = pointer to current longword in FAOSTK
                           0B87  ;         R6 = address of CDB for class to display
                           0B87  ;         R9  = scratch
                           0B87  ;         R10 = address of buffer block
                           0B87  ;         R11 = address of TM4, a temporary stack area
                           0B87  ;
                           0B87  ;         Other registers: see below
                           0B87  ;
                           0B87  ;--
                           0B87
                           0B87
                     0FFC  0B87  .ENTRY  FILL_DISP_BUFF, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           0B89
     56    04 BC     D0    0B89          MOVL    @4(AP),R6                ; Load CDB pointer
                           0B8D          ALLOC   TM4$K_SIZE,R0,R11        ; Allocate local temp storage
                           0B9A
     5A    2E A6     D0    0B9A          MOVL    CDB$A_BUFFERS(R6),R10    ; Load address of buffer block (MBP)
  0E 4B A6     05    E1    0B9E          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if not homog class
     50    32 A6     D0    0BA3          MOVL    CDB$A_CDX(R6),R0         ; Get CDX address
     50    07 A0     9A    0BA7          MOVZBL  CDX$B_IDISCONSEC(R0),R0  ; Get no. of curr display item
                 50  D7    0BAB          DECL    R0                       ; Decrement to use as index
     5A      6A40    D0    0BAD          MOVL    (R10)[R0],R10            ; Get MBP ptr for homog class
                           0BB1
                           0BB1  ;
                           0BB1  ; Return time stamp from most recent collection buffer to caller.
                           0BB1  ;
                           0BB1
                           0BB1  10$:
        59    6A     D0    0BB1          MOVL    MBP$A_BUFFERA(R10),R9    ; Assume BufferA is current
  04 4B A6     01    E1    0BB4          BBC     #CDB$V_SWAPBUF,CDB$L_FLAGS(R6),20$ ; Br if so
     59    04 AA     D0    0BB9          MOVL    MBP$A_BUFFERB(R10),R9    ; BufferB is current
                           0BBD  20$:
  08 BC    03 A9     7D    0BBD          MOVQ    MNR_CLS$Q_STAMP(R9),@8(AP) ; Give current time stamp to caller
                           0BC2
     6B    18 A6     D0    0BC2          MOVL    CDB$L_ECOUNT(R6), -      ; Pick up element count
                           0BC6                  TM4$L_ECOUNT(R11)
  04 AB    1C A6     D0    0BC6          MOVL    CDB$A_ITMSTR(R6), -      ; ... and item string addr
                           0BCB                  TM4$A_ITMSTR(R11)
                           0BCB
                           0BCB  ;
                           0BCB  ; Note -- at this point, R10 contains MBP pointer; element count
                           0BCB  ; and item string address are in TM4$L_ECOUNT and TM4$A_ITMSTR.
                           0BCB  ;
                           0BCB
  03 4B A6     04    E1    0BCB          BBC     #CDB$V_STD,CDB$L_FLAGS(R6),50$ ; Branch if non-standard class
                           0BD0
               0100  31    0BD0          BRW     FDB_STD                  ; Go process standard class
                           0BD3
                           0BD3  50$:                                     ; Non-standard class (PROCESSES)
     00    42 A6     91    0BD3          CMPB    CDB$B_ST(R6),#REG_PROC   ; Regular PROCESSES display?
                 37  13    0BD7          BEQL    FDB_REGPROC              ; Yes -- go fill display buffer for it
                           0BD9                                           ; No -- TOP PROCESSES display
                           0BD9
                           0BD9  ;
                           0BD9  ; Calculate the two quantites BPU and GMIN for use later in computing
```

```
                                  0BD9   3408 ; the size of the bar graph:
                                  0BD9   3409 ;
                                  0BD9   3410 ;       BPU  - floating longword, no of bar chars per unit of output value
                                  0BD9   3411 ;       GMIN - integer longword, min value which graph can represent for this class
                                  0BD9   3412 ;
                                  0BD9   3413
         0E 45 A6    00      E0   0BD9   3414         BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),60$ ; Check for percent requested
              57  3C A6      4E   0BDE   3415         CVTLF   CDB$L_RANGE(R6),R7          ; No percent -- get floating range for graph
     0000000A'EF  38 A6      D0   0BE2   3416         MOVL    CDB$L_MIN(R6),GMIN          ; Get minimum value for graph
                     0D      11   0BEA   3417         BRB     70$                         ; Join common code
                                  0BEC   3418 60$:
         57     00000064 8F  4E   0BEC   3419         CVTLF   #100,R7                     ; 100 is range of percent graph
                0000000A'EF  D4   0BF3   3420         CLRL    GMIN                        ; 0 is min value of percent graph
                                  0BF9   3421 70$:
                  58     28  4C   0BF9   3422         CVTBF   #MAXBARS,R8                 ; Get max bar chars per line
    00000006'EF   58     57  47   0BFC   3423         DIVF3   R7,R8,BPU                   ; Calculate bar chars per unit of output
                                  0C04   3424
                         56  DD   0C04   3425         PUSHL   R6                          ; Stack PROCESSES CDB pointer
       000012A7'EF       01  FB   0C06   3426         CALLS   #1,FILL_TOP                 ; Fill display buffer for TOP display
                       025B  31   0C0D   3427         BRW     FDB_RET                     ; ... and go return
```

```
                            0C10  ;
                            0C10  ; Fill display buffer for non-standard class (PROCESSES)
                            0C10  ; (regular display).
                            0C10  ;
                            0C10  ; Register usage:
                            0C10  ;
                            0C10  ;           R0 = scratch
                            0C10  ;           R1 = scratch
                            0C10  ;           R2 = process index
                            0C10  ;           R3 = process count
                            0C10  ;           R5 = pointer to current longword in FAOSTK
                            0C10  ;           R6 = address of CDB for class to display
                            0C10  ;           R7 = scratch
                            0C10  ;           R9 = CURRENT collection buffer
                            0C10  ;           R10 = address of buffer block
                            0C10  ;           R11 = address of TM4, a temporary stack area
                            0C10  ;
                            0C10
                            0C10  FDB_REGPROC:
                            0C10
        59    0D      C0    0C10          ADDL2   #MNR_CLS$K_HSIZE,R9     ; Point to PROCESSES prefix
       53   04 A9     D0    0C13          MOVL    MNR_PRO$L_PCTINT(R9),R3 ; Get process count
 50    00000000'EF    D0    0C17          MOVL    MCAPTR,R0               ; Get MCA pointer
       18 A0   53     D0    0C1E          MOVL    R3,MCA$L_PROC_DISP(R0)  ; Save count for the display rtn
        59    08      C0    0C22          ADDL2   #MNR_PRO$K_PSIZE,R9     ; Point to first data block
        52    01      D0    0C25          MOVL    #1,R2                   ; Init loop counter
       55   08 AA     D0    0C28          MOVL    MBP$A_PR_FAOSTK(R10),R5 ; Init FAO stack (display buffer) pointer
                            0C2C
                            0C2C  ;
                            0C2C  ; Move individual items for this process from current data block
                            0C2C  ; in collection buffer to longwords in FAO stack.
                            0C2C
                            0C2C  10$:
                            0C2C
        50    69      D0    0C2C          MOVL    MNR_PRO$L_IPID(R9),R0   ; Pick up internal PID
       20 A6   33     B1    0C2F          CMPW    #MNR_PRO$C_EPID,CDB$W_BLKLEN(R6) ; See if we have an EPID
              04      18    0C33          BGEQ    15$                     ; Br if we do not
       50   33 A9     D0    0C35          MOVL    MNR_PRO$L_EPID(R9),R0   ; Take the EPID instead
                            0C39  15$:
        85    50      D0    0C39          MOVL    R0,(R5)+                ; PID to FAO stack
                            0C3C
                            0C3C  ;
                            0C3C  ; Get STATE cstring pointer
                            0C3C  ;
                            0C3C
       50   08 A9     3C    0C3C          MOVZWL  MNR_PRO$W_STATE(R9),R0  ; Get state number
                            0C40
 50   00000000'EF40   D0    0C40          MOVL    STATELIST[R0],R0        ; Get the STATE cstring.
              02      B1    0C48          CMPW    #SCH$C_MWAIT,-          ; Is the process in MWAIT?
           08 A9            0C4A                  MNR_PRO$W_STATE(R9)     ;
              2B      12    0C4C          BNEQU   20$                     ;  No, keep process' STATE cstring.
              37      B1    0C4E          CMPW    #MNR_PRO$L_EFWM,-       ;  Yes, see if we have an EFWM
           20 A6            0C50                  CDB$W_BLKLEN(R6)
              25      18    0C52          BGEQ    20$                     ;  No, simply use MWAIT cstring
 50    00000000'EF    D0    0C54          MOVL    MWAITLIST,R0            ;  Yes, get the generic MUTEX cstring.
              1F      E0    0C5B          BBS     #31,-                   ; Is this is a MUTEX address?
        19 37 A9            0C5D                  MNR_PRO$L_EFWM(R9),20$  ;  Yes, keep the MUTEX cstring.
```

```
      50   00000000'EF   D0  0C60  3486          MOVL    RWAITLIST,R0              ; No, get the generic RWUNK cstring.
                    0F   B1  0C67  3487          CMPW    #RSN$_MAX,-               ; Is this resource wait code defined?
                 37 A9       0C69  3488                  MNR_PRO$L_EFWM(R9)        ;
                    0C   1B  0C6B  3489          BLEQU   20$                       ; No, keep the RWUNK cstring.
            50   37 A9   3C  0C6D  3490          MOVZWL  MNR_PRO$L_EFWM(R9),R0     ; Yes, get the resource wait number.
    50   00000000'EF40   D0  0C71  3491          MOVL    RWAITLIST[R0],R0          ; Get the RWccc cstring.
                             0C79  3492
                             0C79  3493 20$:
               85    50  D0  0C79  3494          MOVL    R0,(R5)+                  ; ... and move it to FAO stack
```

```
                                0C7C  3496 ;
                                CC7C  3497 ; Get remainder of process items
                                0C7C  3498 ;
                                0C7C  3499
            57    0A A9    9A   0C7C  3500          MOVZBL  MNR_PRO$B_PRI(R9),R7      ; Get priority (31's complement)
         85   1F    57     C3   0C80  3501          SUBL3   R7,#31,(R5)+              ; Complement it and move to FAO stack
            85    0B A9    7D   0C84  3502          MOVQ    MNR_PRO$O_LNAME(R9),(R5)+ ; Process name cstring to FAO stack
            85    13 A9    7D   0C88  3503          MOVQ    MNR_PRO$O_LNAME+8(R9),(R5)+ ; .....
            85    F0 A5    9A   0C8C  3504          MOVZBL  -16(R5),(R5)+             ; Length of process name to FAO stack
         85   55    13     C3   0C90  3505          SUBL3   #19,R5,(R5)+              ; Address of process name to FAO stack
            85    1B A9    3C   0C94  3506          MOVZWL  MNR_PRO$W_GPGCNT(R9),(R5)+ ; Global page count to FAO stack
    57   1D A9    1B A9    A1   0C98  3507          ADDW3   MNR_PRO$W_GPGCNT(R9),MNR_PRO$W_PPGCNT(R9),R7
                                0C9E  3508                                            ; Get sum of global & process page cnts
               85   57     3C   0C9E  3509          MOVZWL  R7,(R5)+                  ; ... and move to FAO stack
         07 1F A9   00     E0   0CA1  3510          BBS     #PCB$V_RES,MNR_PRO$L_STS(R9),30$ ; Br if process was resident
               55   10     C0   0CA6  3511          ADDL2   #16,R5                    ; Process non-res; skip next 4 longwords
                    85     D4   0CA9  3512          CLRL    (R5)+                     ; Clear CPUTIM ptr to indicate non-res
                    16     11   0CAB  3513          BRB     40$                       ; ... and continue
                                0CAD  3514 30$:                                       ; Resident process
            85    23 A9    D0   0CAD  3515          MOVL    MNR_PRO$L_DIOCNT(R9),(R5)+   ; DIO count to FAO stack
            85    27 A9    D0   0CB1  3516          MOVL    MNR_PRO$L_PAGEFLTS(R9),(R5)+ ; Page fault count to FAO stack
85    00    2B A9   000186A0 8F    7A   0CB5  3517          EMUL    #100000,MNR_PRO$L_CPUTIM(R9),#0,(R5)+
                                0CBF  3518                                            ; Xlate ticks to quad time val & move to FAO
         85   55    08     C3   0CBF  3519          SUBL3   #8,R5,(R5)+               ; ... and move ptr to it into FAO stack
                                0CC3  3520 40$:
            50    20 A6    3C   0CC3  3521          MOVZWL  CDB$W_BLKLEN(R6),R0       ; Get length of a data block
               59   50     C0   0CC7  3522          ADDL2   R0,R9                     ; Point to next (process) data block in coll
      FF5C 52   01   53    F1   0CCA  3523          ACBL    R3,#1,R2,10$              ; Loop to move next process to FAO stack
                  0198     31   0CD0  3524          BRW     FDB_RET                   ; All processes done ... go return
```

```
                                  0CD3  ;
                                  0CD3  ; Fill display buffer (FAOSTK) for standard classes.
                                  0CD3  ;
                                  0CD3  ; Register usage:
                                  0CD3  ;
                                  0CD3  ;       R0 = scratch, address of current item number
                                  0CD3  ;       R1 = scratch
                                  0CD3  ;       R2 = data item index
                                  0CD3  ;       R3 = address of statistics buffer from buffer block
                                  0CD3  ;       R4 = address of IDB for current data item
                                  0CD3  ;       R5 = pointer to current longword in FAOSTK
                                  0CD3  ;       R6 = address of CDB for class to display
                                  0CD3  ;       R7 = scratch
                                  0CD3  ;       R8 = statistic code (ALL =0, CUR=1, AVE=2, MIN=3, MAX=4)
                                  0CD3  ;       R9 = scratch
                                  0CD3  ;       R10 = address of buffer block
                                  0CD3  ;       R11 = address of TM4, a temporary stack area
                                  0CD3  ;
                                  0CD3
                                  0CD3 FDB_STD:
                                  0CD3
                                  0CD3  ;
                                  0CD3  ; For homogeneous class, store number of elements to display
                                  0CD3  ; (for use in later display routines).
                                  0CD3  ;
                                  0CD3
          08 4B A6     05     E1  0CD3        BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if a heterogeneous class
                                  0CD8
            50   32 A6     D0     0CD8        MOVL    CDB$A_CDX(R6),R0        ; Get CDB extension for HOMOG class
            1C A0     6B   D0     0CDC        MOVL    TM4$L_ECOUNT(R11), -    ; Store number of elts to display
                                  0CE0                CDX$L_DCOUNT(R0)
                                  0CE0 10$:
                      6B   D5     0CE0        TSTL    TM4$L_ECOUNT(R11)       ; Any elements to display?
                      03   12     0CE2        BNEQ    20$                     ; Br if have some
                    0184   31     0CE4        BRW     FDB_RET                 ; Else go exit if none
                                  0CE7 20$:
                                  0CE7
                                  0CE7  ;
                                  0CE7  ; If AVERAGE or ALL statistic requested, calculate floating elapsed
                                  0CE7  ; seconds from start of monitor request to time of most recent collection
                                  0CE7  ; event. Store it on the stack for later use.
                                  0CE7  ;
                                  0CE7
            00   42 A6     91     0CE7        CMPB    CDB$B_ST(R6),#ALL_STAT  ; ALL stats requested ?
                      06   13     0CEB        BEQL    30$                     ; Br if yes
            02   42 A6     91     0CED        CMPB    CDB$B_ST(R6),#AVE_STAT  ; Was AVERAGE stat requested ?
                      29   12     0CF1        BNEQ    40$                     ; No -- skip following calculation
                                  0CF3 30$:
       57  00000000'EF     D0     0CF3        MOVL    MRBPTR,R7               ; Get temp pointer to MRB
            51    07 A9    D0     0CFA        MOVL    MNR_CLS$Q_STAMP+4(R9),R1 ; Get high order bits of time
        50   03 A9   67    C3     0CFE        SUBL3   MRB$Q_BEGINNING(R7),MNR_CLS$Q_STAMP(R9),R0 ; Compute elapsed time si
            51    04 A7    D9     0D03        SBWC    MRB$Q_BEGINNING+4(R7),R1 ; Get high order difference
51    50    50  00002710 8F  7B   0D07        EDIV    #10000,R0,R0,R1         ; Turn time into milliseconds
                 50   50   4E     0D10        CVTLF   R0,R0                   ; Floating milliseconds
  0C AB    50  0000457A 8F   47   0D13        DIVF3   #1000,R0,TM4$L_FLTSECS(R11) ; Save floating seconds on stack
                                  0D1C                                        ; ... fcr later use
```

```
                               0D1C  3583 ;
                               0D1C  3584 ; Execute special routines if this is the SYSTEM class
                               0D1C  3585 ;
                               0D1C  3586
                               0D1C  3587 40$:
      39 4B A6    08     E1    0D1C  3588          BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),55$ ; Br if not SYSTEM class
        00    42 A6     91    0D21  3589          CMPB    CDB$B_ST(R6),#ALL_STAT   ; ALL stats requested ?
                  18     13    0D25  3590          BEQL    50$                      ; Br if yes
                               0D27  3591          ALLOC   <4*ECOUNT_SYS_SINGLE>,R0,R3 ; Single stat -- get dummy buffer
              0265     31    0D3C  3592          BRW     FDB_SYS_SINGLE           ; ... and branch to special rtn
                               0D3F  3593 50$:
                               0D3F  3594          ALLOC   <MBP$K_SIZE+<4*4*ECOUNT_SYS_ALL>>,R0,R10
                               0D54  3595                                           ; Alloc dummy MBP and stats buffers
                               0D54  3596                                           ; ... pointed to by R10
         00001209'EF      16    0D54  3597          JSB     FDB_SYS_ALL              ; Pre-process the SYSTEM class
                               0D5A  3598 55$:
                               0D5A  3599
    55   00000103'EF      DE    0D5A  3600          MOVAL   FAOSTK,R5                ; Load addr of display buffer
        58    42 A6     9A    0D61  3601          MOVZBL  CDB$B_ST(R6),R8          ; Load requested statistic
                               0D65  3602
                               0D65  3603 ;
                               0D65  3604 ; If single statistic (not ALL) requested, calculate and store two quantities
                               0D65  3605 ; for later use:
                               0D65  3606 ;
                               0D65  3607 ;        BPU  - floating longword, no of bar chars per unit of output value
                               0D65  3608 ;        GMIN - integer longword, min value which graph can represent for this class
                               0D65  3609 ;
                               0D65  3610
            00      58     D1    0D65  3611          CMPL    R8,#ALL_STAT             ; ALL requested?
                    2B     13    0D68  3612          BEQL    80$                      ; Yes -- continue
      0E 45 A6      00     E0    0D6A  3613          BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),60$ ; No -- check for percent reques
          50     3C A6     4E    0D6F  3614          CVTLF   CDB$L_RANGE(R6),R0       ; No percent -- get floating range for graph
  0000000A'EF    38 A6     D0    0D73  3615          MOVL    CDB$L_MIN(R6),GMIN       ; Get minimum value for graph
                    0D     11    0D7B  3616          BRB     70$                      ; Join common code
                               0D7D  3617 60$:
    50   00000064 8F      4E    0D7D  3618          CVTLF   #100,R0                  ; 100 is range of percent graph
         0000000A'EF      D4    0D84  3619          CLRL    GMIN                     ; 0 is min value of percent graph
                               0D8A  3620 70$:
            51      28     4C    0D8A  3621          CVTBF   #MAXBARS,R1              ; Get max bar chars per line
 00000006'EF    51      50     47    0D8D  3622          DIVF3   R0,R1,BPU                ; Calculate bar chars per unit of output
                               0D95  3623
                               0D95  3624 ;
                               0D95  3625 ; For homogeneous class, determine item type for use below.
                               0D95  3626 ;
                               0D95  3627
                               0D95  3628 80$:
      1C 4B A6      05     E1    0D95  3629          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),90$ ; Br if a heterogeneous class
                               0D9A  3630
          52     32 A6     D0    0D9A  3631          MOVL    CDB$A_CDX(R6),R2         ; Get CDB extension for HOMOG class
                               0D9E  3632
          53     08 A2     9A    0D9E  3633          MOVZBL  CDX$B_IDISINDEX(R2),R3   ; Get item index for this disp event
       54     04 BB43     9A    0DA2  3634          MOVZBL  @TM4$A_ITMSTR(R11)[R3],R4 ; Load IDB item number
              54     11     C4    0DA7  3635          MULL2   #IDB$K_ILENGTH,R4        ; Compute index into IDB table
      54    0000'CF44     9E    0DAA  3636          MOVAB   W^PERFTABLE[R4],R4       ; Address of IDB for this item
    00E0'CF      0A A4     B0    0DB0  3637          MOVW    IDB$W_TYPE(R4),W^ITEM_TYPE ; Save item type for use below
```

```
                        0DB6  ;
                        0DB6  ; Loop once for each element in this class. Pick up transformed value for
                        0DB6  ; desired statistic from appropriate buffer within collection buffer
                        0DB6  ; block. Do computation on transformed value if required, and place
                        0DB6  ; whole and fractional portions of result into display buffer (FAOSTK).
                        0DB6  ;
                        0DB6
                        0DB6  90$:
                6B  D7  0DB6          DECL    TM4$L_ECOUNT(R11)         ; Set up number of elements to
                        0DB8                                            ; ... display as a loop limit
       50    04 AB  D0  0DB8          MOVL    TM4$A_ITMSTR(R11),R0      ; Address of item-number string
                52  D4  0DBC          CLRL    R2                        ; Clear element index register
                        0DBE  FDB_BEG:
    12 4B A6    05  E0  0DBE          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if a homogeneous class
          54    80  9A  0DC3          MOVZBL  (R0)+,R4                  ; Get next item number
          54    11  C4  0DC6          MULL2   #IDB$K_ILENGTH,R4         ; Compute index into IDB table
   54   0000'CF44   9E  0DC9          MOVAB   W^PERFTABLE[R4],R4        ; Address of IDB for this item
 00E0'CF     0A A4  B0  0DCF          MOVW    IDB$W_TYPE(R4),W^ITEM_TYPE ; Save item type for use below
                        0DD5  10$:
                        0DD5          CASE    R8,<F_ALL,F_CUR,F_AVE,F_MIN,F_MAX>,L ; Select on requested statistic
                        0DE3
                        0DE3  F_ALL:
                        0DE3  F_CUR:
    09 45 A6    00  E0  0DE3          BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$ ; Br if percent requested
       53    08 AA  D0  0DE8          MOVL    MBP$A_STATS(R10),R3       ; Load addr of STATS buffer
              0104  30  0DEC          BSBW    INTORFL                   ; Process an integer or floating value
                0A  11  0DEF          BRB     20$                       ; ... and continue
                        0DF1  10$:
       53    18 AA  D0  0DF1          MOVL    MBP$A_PCSTATS(R10),R3     ; Load addr of PCSTATS buffer
        59    6342  D0  0DF5          MOVL    (R3)[R2],R9               ; Load tenths of % value
                74  10  0DF9          BSB     PCTEN                     ; Process a tenths of % value
                        0DFB  20$:
                58  D5  0DFB          TSTL    R8                        ; Was ALL statistic requested?
                5D  12  0DFD          BNEQ    COMMON                    ; No -- join common code
                        0DFF                                            ; Yes -- continue to AVE
                        0DFF  F_AVE:
    09 45 A6    00  E0  0DFF          BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$ ; Br if percent requested
       53    14 AA  D0  0E04          MOVL    MBP$A_SUM(R10),R3         ; Load addr of SUM buffer
              007B  30  0E08          BSBW    INTAVE                    ; Process an integer average value
                17  11  0E0B          BRB     20$                       ; ... and continue
                        0E0D  10$:
       53    24 AA  D0  0E0D          MOVL    MBP$A_PCSUM(R10),R3       ; Load addr of PCSUM buffer
  51   00000000'EF  D0  0E11          MOVL    MCAPTR,R1                 ; Get MCA pointer
   57    0C A1  01  C3  0E18          SUBL3   #1,MCA$L_COLLCNT(R1),R7   ; Get no of colls, don't count 1st
    59    6342  57  C7  0E1D          DIVL3   R7,(R3)[R2],R9            ; Get average tenths % value
                4B  10  0E22          BSB     PCTEN                     ; ... and process it
                        0E24  20$:
                58  D5  0E24          TSTL    R8                        ; Was ALL statistic requested?
                34  12  0E26          BNEQ    COMMON                    ; No -- join common code
                        0E28                                            ; Yes -- continue to MIN
                        0E28  F_MIN:
    09 45 A6    00  E0  0E28          BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$ ; Br if percent requested
       53    0C AA  D0  0E2D          MOVL    MBP$A_MIN(R10),R3         ; Load addr of MIN buffer
              00BF  30  0E31          BSBW    INTORFL                   ; Process an integer or floating value
                0A  11  0E34          BRB     20$                       ; ... and continue
                        0E36  10$:
       53    1C AA  D0  0E36          MOVL    MBP$A_PCMIN(R10),R3       ; Load addr of PCMIN buffer
```

```
            59    6342   D0  0E3A  3696           MOVL    (R3)[R2],R9                  ; Load tenths of % value
                    2F   10  0E3E  3697           BSB     PCTEN                        ; Process a tenths of % value
                             0E40  3698 20$:
                    58   D5  0E40  3699           TSTL    R8                           ; Was ALL statistic requested?
                    18   12  0E42  3700           BNEQ    COMMON                       ; No -- join common code
                             0E44  3701                                                ; Yes -- continue to MAX
                             0E44  3702 F_MAX:
     09 45 A6       00   E0  0E44  3703           BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$ ; Br if percent requested
         53    10 AA     D0  0E49  3704           MOVL    MBP$A_MAX(R10),R3            ; Load addr of MAX buffer
                  00A3   30  0E4D  3705           BSBW    INTORFL                      ; Process an integer or floating value
                    0A   11  0E50  3706           BRB     COMMON                       ; ... and continue
                             0E52  3707 10$:
         53    20 AA     D0  0E52  3708           MOVL    MBP$A_PCMAX(R10),R3          ; Load addr of PCMAX buffer
            59    6342   D0  0E56  3709           MOVL    (R3)[R2],R9                  ; Load tenths of % value
                    13   10  0E5A  3710           BSB     PCTEN                        ; Process a tenths of % value
                             0E5C  3711
                             0E5C  3712 COMMON:                                        ; Common return point from CASE
     04 10 A4       00   E1  0E5C  3713           BBC     #IDB$V_PCNT,IDB$B_FLAGS(R4),10$ ;branch if it is not a pcnt
                    50   D6  0E61  3714           INCL    R0                           ; increment index into item table
                    52   D6  0E63  3715           INCL    R2                           ; increment index into data buffer
                             0E65  3716 10$:
FF53 52       01    6B   F1  0E65  3717           ACBL    TM4$L_ECOUNT(R11),#1,R2,FDB_BEG ; Loop once for each element
                             0E6B  3718
                             0E6B  3719 FDB_RET:
              50    01   D0  0E6B  3720           MOVL    #SS$_NORMAL,R0               ; Indicate success
                         04  0E6E  3721           RET                                  ; ... and return
```

```
                                 0E6F   ;
                                 0E6F   ; PCTEN - Transform an integer tenths of percent value to a whole integer
                                 0E6F   ;               percent value and an optional integer tenths "remainder"
                                 0E6F   ;               value. Place result(s) in display buffer (FAOSTK).
                                 0E6F   ;
                                 0E6F   ;               R2 = data item index
                                 0E6F   ;               R3 = address of source statistics buffer
                                 0E6F   ;               R5 = address of current longword in display buffer
                                 0E6F   ;               R7 = scratch
                                 0E6F   ;               R8 = code for requested statistic
                                 0E6F   ;               R9 = tenths of % value (input)
                                 0E6F   ;
                                 0E6F
                                 0E6F   PCTEN:
        85    59    0A    C7     0E6F           DIVL3   #10,R9,(R5)+            ; Compute whole percent value
                    58    D5     0E73           TSTL    R8                      ; Requested ALL statistics?
                    05    13     0E75           BEQL    10$                     ; Yes -- continue
                  00B4    30     0E77           BSBW    CALC_BAR                ; No -- go do bar graph calcs
                    09    11     0E7A           BRB     20$                     ; ... and get out
                                 0E7C   10$:
    57     FC A5    0A    C5     0E7C           MULL3   #10,-4(R5),R7           ; Need fraction for tabular display
        85    59    57    C3     0E81           SUBL3   R7,R9,(R5)+             ; Into display buffer
                                 0E85   20$:
                          05     0E85           RSB
                                 0E86
                                 0E86   ;
                                 0E86   ; INTAVE - Transform an integer sum of level values or counts into an
                                 0E86   ;               average size/collection or rate/second. Place whole
                                 0E86   ;               and optional fractional parts into display buffer (FAOSTK).
                                 0E86   ;
                                 0E86   ;               R1 = scratch
                                 0E86   ;               R2 = data item index
                                 0E86   ;               R3 = address of source statistics buffer
                                 0E86   ;               R4 = address of IDB for current item
                                 0E86   ;               R5 = address of current longword in display buffer
                                 0E86   ;               R7 = scratch
                                 0E86   ;               R8 = code for requested statistic
                                 0E86   ;               R9 = scratch
                                 0E86   ;               R11 = address of TM4
                                 0E86   ;               TM4$L_FLTSECS = floating seconds from start of monitor request
                                 0E86   ;                               to most recent collection event.
                                 0E86   ;
                                 0E86
                                 0E86   INTAVE:
      25 10 A4    00      E1     0E86           BBC     #IDB$V_PCNT,IDB$B_FLAGS(R4),7$  ;branch if it is not a percent
                                 0E8B   ;
                                 0E8B   ;If it is percent,  compute the average based on the sums of item1 and item2
                                 0E8B   ;
                    65    D4     0E8B           CLRL    (R5)                    ;Zero the current longword in Fao buff
57    00000064 8F  04 A342 C5    0E8D           MULL3   4(R3)[R2],#100,R7       ;sum for item1*100 into R7
              57    57    4E     0E97           CVTLF   R7,R7                   ;convert to float
        59     08 A342    D0     0E9A           MOVL    8(R3)[R2],R9            ;sum for (item1+item2) into R9
                    59    D5     0E9F           TSTL    R9                      ;item1 + item2 = 0?
                    09    13     0EA1           BEQL    5$                      ;skip divide if so
              59    59    4E     0EA3           CVTLF   R9,R9                   ;convert to float
              57    59    46     0EA6           DIVF    R9,R7                   ;compute floating avg
              65    57    4A     0EA9           CVTFL   R7,(R5)                 ;stack whole part for fao
```

```
                              0EAC  3780 5$:
                    85    D5  0EAC  3781         TSTL     (R5)+                       ;increment display buff pointer
                    28    11  0EAE  3782         BRB      25$                         ;and move on
                              0EB0  3783
                              0EB0  3784 7$:
             57   6342    4E  0EB0  3785         CVTLF    (R3)[R2],R7                 ; Get floating sum
    0000'8F    00E0'CF    B1  0EB4  3786         CMPW     W^ITEM_TYPE,#COUNT_TYPE     ; This item a count?
                    06    12  0EBB  3787         BNEQ     10$                         ; No -- assume level type
             57    0C AB  46  0EBD  3788         DIVF     TM4$L_FLTSECS(R11),R7       ; Yes -- get floating avg rate/second
                    12    11  0EC1  3789         BRB      20$                         ; ... and continue
                              0EC3  3790 10$:
     51   00000000'EF     D0  0EC3  3791         MOVL     MCAPTR,R1                   ; Get MCA pointer
      59   0C A1    01    C3  0ECA  3792         SUBL3    #1,MCA$L_COLLCNT(R1),R9     ; Get no of colls, don't count 1st
                 59    59  4E  0ECF  3793         CVTLF    R9,R9                       ; Get floating no of collections
                 57    59  46  0ED2  3794         DIVF     R9,R7                       ; Compute floating avg size/collection
                              0ED5  3795 20$:
                 85    57  4A  0ED5  3796         CVTFL    R7,(R5)+                    ; Stack whole part for fao
                              0ED8  3797 25$:
                    58    D5  0ED8  3798         TSTL     R8                          ; Requested ALL statistics?
                    05    13  0EDA  3799         BEQL     30$                         ; Yes -- continue
                  004F    30  0EDC  3800         BSBW     CALC_BAR                    ; No -- go do bar graph calcs
                    11    11  0EDF  3801         BRB      40$                         ; ... and get out
                              0EE1  3802 30$:
             59    FC A5  4E  0EE1  3803         CVTLF    -4(R5),R9                   ; Get back truncated part
                 57    59  42  0EE5  3804         SUBF     R9,R7                       ; Compute fraction to two ...
     57   000043C8 8F     44  0EE8  3805         MULF     #100,R7                     ; ... digits for tabular display
                 85    57  4A  0EEF  3806         CVTFL    R7,(R5)+                    ; Stack fraction for fao
                              0EF2  3807 40$:
                          05  0EF2  3808         RSB
```

```
                                0EF3
                                0EF3  ;
                                0EF3  ; INTORFL - Place whole and optional fractional parts of integer value
                                0EF3  ;                 (level) or floating rate/second value (count) into
                                0EF3  ;                 display buffer (FAOSTK).
                                0EF3  ;
                                0EF3  ;                 R2 = data item index
                                0EF3  ;                 R3 = address of source statistics buffer
                                0EF3  ;                 R4 = address of IDB for current item
                                0EF3  ;                 R5 = address of current longword in display buffer
                                0EF3  ;                 R7 = scratch
                                0EF3  ;                 R8 = code for requested statistic
                                0EF3  ;                 R9 = scratch
                                0EF3  ;
                                0EF3
                                0EF3 INTORFL:
 0000'8F    00E0'CF     B1      0EF3          CMPW    W^ITEM_TYPE,#COUNT_TYPE ; Is this item a count?
                 11     13      0EFA          BEQL    20$                     ; Br if yes
         85    6342     D0      0EFC          MOVL    (R3)[R2],(R5)+          ; Move level value to disp buffer
                 58     D5      0F00          TSTL    R8                      ; ALL statistics requested?
                 05     13      0F02          BEQL    10$                     ; Yes -- continue
               0027     30      0F04          BSBW    CALC_BAR                ; No -- go do bar graph calcs
                 24     11      0F07          BRB     40$                     ; ... and get out
                                0F09 10$:
                 85     D4      0F09          CLRL    (R5)+                   ; Stack fractional part
                 20     11      0F0B          BRB     40$                     ; ... and exit
                                0F0D 20$:
         85    6342     4A      0F0D          CVTFL   (R3)[R2],(R5)+          ; Stack whole part of rate (for count)
                 58     D5      0F11          TSTL    R8                      ; ALL statistics requested?
                 05     13      0F13          BEQL    30$                     ; Yes -- continue
               0016     30      0F15          BSBW    CALC_BAR                ; No -- go do bar graph calcs
                 13     11      0F18          BRB     40$                     ; ... and get out
                                0F1A 30$:
           57  FC A5    4E      0F1A          CVTLF   -4(R5),R7               ; Get back rounded part
      59   6342  57     43      0F1E          SUBF3   R7,(R3)[R2],R9          ; Compute fraction
   59   000043C8 8F     44      0F23          MULF    #100,R9                 ; ... to two digits
            85   59     4A      0F2A          CVTFL   R9,(R5)+                ; Stack fractional part
                                0F2D 40$:
                        05      0F2D          RSB                             ; Return to caller
```

```
                                 0F2E  ;
                                 0F2E  ; CALC_BAR - Replace integer longword value in display buffer with three
                                 0F2E  ;                longwords representing the width of a field to display
                                 0F2E  ;                the value (0 or 7), the value, and the number of bar
                                 0F2E  ;                characters needed to represent the value in a bar graph.
                                 0F2E  ;
                                 0F2E  ;           Note -- if SYSTEM class, merely annex bar count to count
                                 0F2E  ;                   already in display buffer.
                                 0F2E  ;
                                 0F2E  ;        Register Inputs:
                                 0F2E  ;
                                 0F2E  ;                R1 = scratch
                                 0F2E  ;                R5 = address of current longword in display buffer;
                                 0F2E  ;                     will be updated to next available longword on output
                                 0F2E  ;                R6 = addr of CDB
                                 0F2E  ;                R7 = scratch
                                 0F2E  ;                R9 = scratch
                                 0F2E  ;
                                 0F2E  ;        Implicit Inputs:
                                 0F2E  ;
                                 0F2E  ;                GMIN - Integer longword, min value for graph
                                 0F2E  ;                BPU  - Floating longword, bar chars per unit of value
                                 0F2E  ;
                                 0F2E
                                 0F2E
                                 0F2E  ;
                                 0F2E  ; This subroutine creates 3 longwords in the FAOSTK array for the current
                                 0F2E  ; data item. The current item is represented by a longword integer value
                                 0F2E  ; which will be displayed to the left of the bar in the bar graph for
                                 0F2E  ; the current class. The current item has already been placed in the
                                 0F2E  ; display buffer by the calling routine; R5 has already been advanced to
                                 0F2E  ; the next available longword. This subroutine must replace the value
                                 0F2E  ; longword in the display buffer (FAOSTK) with 3 longwords, leaving R5
                                 0F2E  ; pointing to the next available longword. The three longwords are:
                                 0F2E  ; width of value field (0 or 7), value, and number of bar characters
                                 0F2E  ; needed to represent the value. The number of bargraph characters is
                                 0F2E  ; computed in floating point and then truncated.
                                 0F2E  ;
                                 0F2E
                                 0F2E CALC_BAR:
       35 4B A6     08      E0   0F2E         BBS     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),30$ ; Br if SYSTEM class
                    57      D4   0F33         CLRL    R7                      ; Assume value field width will be 0
          65    FC A5      D0   0F35         MOVL    -4(R5),(R5)             ; Move value ahead in display buffer
                    03      13   0F39         BEQL    10$                     ; If value zero, go move 0 field width
             57     07      9A   0F3B         MOVZBL  #7,R7                   ; Value is non-zero; field width is 7
                                 0F3E 10$:
          FC A5     57      D0   0F3E         MOVL    R7,-4(R5)               ; Move value field width into display buffer
                                 0F42  ;
                                 0F42  ; Now calculate number of bars to output
                                 0F42  ;
                    57      D4   0F42         CLRL    R7                      ; Assume no bars will be output
   59    85   0000000A'EF   C3   0F44         SUBL3   GMIN,(R5)+,R9           ; Calc units of value to output and ...
                                 0F4C                                         ; ... advance R5 to "no of bars" longword
                    15      15   0F4C         BLEQ    20$                     ; Output no bars if leq zero
             59     59      4E   0F4E         CVTLF   R9,R9                   ; Convert units to floating
        59    00000006'EF   44   0F51         MULF2   BPU,R9                  ; Bars/unit * units => bars to output
             57     59      4A   0F58         CVTFL   R9,R7                   ; Integer number to output
```

```
                   57     28   D1  0F5B  3907          CMPL    #MAXBARS,R7                 ; Check for upperbound
                          03   18  0F5E  3908          BGEQ    20$                         ; Continue if within range
                   57     28   D0  0F60  3909          MOVL    #MAXBARS,R7                 ; Else make it within range
                                   0F63  3910 20$:
                   85     57   D0  0F63  3911          MOVL    R7,(R5)+                    ; Move number of bars to display buffer ...
                                   0F66  3912                                              ; ... and advance R5 to next longword
                          3B   11  0F66  3913          BRB     50$                         ; Go return to caller
                                   0F68  3914
                                   0F68  3915 ;
                                   0F68  3916 ; Special processing for SYSTEM class
                                   0F68  3917 ;
                                   0F68  3918
                                   0F68  3919 30$:
       51     00000000'EF      DE  0F68  3920          MOVAL   FMT_SYS_SINGLE,R1           ; Get vector of format codes
           00'8F     6142      91  0F6F  3921          CMPB    (R1)[R2],#NUMB_ONLY         ; Number only desired?
                          2D   13  0F74  3922          BEQL    50$                         ; Br if yes -- all done
                                   0F76  3923
                                   0F76  3924 ;
                                   0F76  3925 ; NUMB_BAR type ... number and bar desired. Number is already in stack.
                                   0F76  3926 ; Now calculate number of bars to output.
                                   0F76  3927 ;
                                   0F76  3928
                          57   D4  0F76  3929          CLRL    R7                          ; Assume no bars will be output
                       FC A5   D5  0F78  3930          TSTL    -4(R5)                      ; Zero units of value to output?
                                   0F7B  3931
                          23   15  0F7B  3932          BLEQ    40$                         ; Output no bars if leq zero
                                   0F7D  3933
       51     00000000'EF      DE  0F7D  3934          MOVAL   BU_SYS_SINGLE,R1            ; Get addr of vector of bar ranges
             59      6142      4E  0F84  3935          CVTLF   (R1)[R2],R9                 ; Get floating range for graph
                   51     1A   4C  0F88  3936          CVTBF   #MAXBARS_SYS,R1             ; Get max bar chars per line
                   51     59   46  0F8B  3937          DIVF2   R9,R1                       ; Calculate bar chars per unit of output
                                   0F8E  3938
             59        FC A5   4E  0F8E  3939          CVTLF   -4(R5),R9                   ; Get floating units of value to output
                                   0F92  3940
                   59     51   44  0F92  3941          MULF2   R1,R9                       ; Bars/unit * units => bars to output
                   57     59   4A  0F95  3942          CVTFL   R9,R7                       ; Integer number to output
                   57     1A   D1  0F98  3943          CMPL    #MAXBARS_SYS,R7             ; Check for upperbound
                          03   18  0F9B  3944          BGEQ    40$                         ; Continue if within range
                   57     1A   D0  0F9D  3945          MOVL    #MAXBARS_SYS,R7             ; Else make it within range
                                   0FA0  3946 40$:
                   85     57   D0  0FA0  3947          MOVL    R7,(R5)+                    ; Move number of bars to display buffer ...
                                   0FA3  3948                                              ; ... and advance R5 to next longword
                                   0FA3  3949 50$:
                               05  0FA3  3950          RSB                                 ; Return to caller
```

```
                                 0FA4  ;
                                 0FA4  ; FDB_SYS_SINGLE
                                 0FA4  ;
                                 0FA4  ; Fill a dummy statistics buffer from similar buffers of the MODES
                                 0FA4  ; STATES and SYSTEM classes. Then call INTORFL and INTAVE routines
                                 0FA4  ; to transform the data in the buffer to items on the FAOSTK.
                                 0FA4  ;
                                 0FA4  ; This routine is entered with a direct branch, and branches to
                                 0FA4  ; FDB_RET when done to return to original caller.
                                 0FA4  ;
                                 0FA4  ; Inputs:
                                 0FA4  ;
                                 0FA4  ;         R0 - R2 scratch
                                 0FA4  ;         R3 - address of statistics buffer.
                                 0FA4  ;         R4 - R5 scratch
                                 0FA4  ;         R6  - address of SYSTEM CDB.
                                 0FA4  ;         R7 - R10 scratch
                                 0FA4  ;         R11 - address of TM4, a temporary stack area
                                 0FA4  ;
                                 0FA4
                                 0FA4  FDB_SYS_SINGLE:
                                 0FA4
     50   00000000'8F      D0    0FA4          MOVL    #<CDB$K_SIZE*MODES_CLSNO>,R0 ; Compute offset to MODES CDB
  50    00000000'EF40      9E    0FAB          MOVAB   CDBHEAD[R0],R0               ; ... get its CDB address
       52      2E A0       D0    0FB3          MOVL    CDB$A_BUFFERS(R0),R2         ; ... and MBP ptr for later use
                                 0FB7
     50   00000000'8F      D0    0FB7          MOVL    #<CDB$K_SIZE*STATES_CLSNO>,R0 ; Compute offset to STATES CDB
  50    00000000'EF40      9E    0FBE          MOVAB   CDBHEAD[R0],R0               ; ... get its CDB address
       54      2E A0       D0    0FC6          MOVL    CDB$A_BUFFERS(R0),R4         ; ... and MBP ptr for later use
                                 0FCA
       55      2E A6       D0    0FCA          MOVL    CDB$A_BUFFERS(R6),R5         ; Get same for SYSTEM class
                                 0FCE
       58      42 A6       9A    0FCE          MOVZBL  CDB$B_ST(R6),R8              ; Get requested stat code
                                 0FD2          CASE    R8,<FS_ALL,FS_CUR,FS_AVE,FS_MIN,FS_MAX>,L
                                 0FE0                                               ; Select on requested stat
                                 0FE0
                                 0FE0  FS_ALL:                                      ; Should not occur
                                 0FE0  FS_CUR:
       52      08 A2       D0    0FE0          MOVL    MBP$A_STATS(R2),R2           ; Load addr of STATS buffer for MODES
       54      08 A4       D0    0FE4          MOVL    MBP$A_STATS(R4),R4           ; Load addr of STATS buffer for STATES
       55      08 A5       D0    0FE8          MOVL    MBP$A_STATS(R5),R5           ; Load addr of STATS buffer for SYSTEM
                  2A       11    0FEC          BRB     FS_COMMON                    ; Join common code
                                 0FEE
                                 0FEE  FS_AVE:
       52      14 A2       D0    0FEE          MOVL    MBP$A_SUM(R2),R2             ; Load addr of SUM buffer for MODES
       54      14 A4       D0    0FF2          MOVL    MBP$A_SUM(R4),R4             ; Load addr of SUM buffer for STATES
       55      14 A5       D0    0FF6          MOVL    MBP$A_SUM(R5),R5             ; Load addr of SUM buffer for SYSTEM
                  1C       11    0FFA          BRB     FS_COMMON                    ; Join common code
                                 0FFC
                                 0FFC  FS_MIN:
       52      0C A2       D0    0FFC          MOVL    MBP$A_MIN(R2),R2             ; Load addr of MIN buffer for MODES
       54      0C A4       D0    1000          MOVL    MBP$A_MIN(R4),R4             ; Load addr of MIN buffer for STATES
       55      0C A5       D0    1004          MOVL    MBP$A_MIN(R5),R5             ; Load addr of MIN buffer for SYSTEM
                  0E       11    1008          BRB     FS_COMMON                    ; Join common code
                                 100A
                                 100A  FS_MAX:
       52      10 A2       D0    100A          MOVL    MBP$A_MAX(R2),R2             ; Load addr of MAX buffer for MODES
```

```
         54    10 A4   D0  100E  4009          MOVL    MBP$A_MAX(R4),R4          ; Load addr of MAX buffer for STATES
         55    10 A5   D0  1012  4010          MOVL    MBP$A_MAX(R5),R5          ; Load addr of MAX buffer for SYSTEM
                  00   11  1016  4011          BRB     FS_COMMON                 ; Join common code
                           1018  4012
                           1018  4013  FS_COMMON:
                           1018  4014
                           1018  4015  ;
                           1018  4016  ; Move items of interest from the source buffers to a single
                           1018  4017  ; destination buffer.
                           1018  4018  ;
                           1018  4019  ; *** NOTE ***  This section contains hard-wired offsets. It
                           1018  4020  ;               assumes that the positions of items in the
                           1018  4021  ;               MODES and STATES class do not change.
                           1018  4022  ;               Not currently using the MODES class for the
                           1018  4023  ;               single display, however.
                           1018  4024  ;
                           1018  4025
                           1018  4026  ;
                           1018  4027  ; *** NOTE ***  R2 has the address of the MODES buffer, but it is
                           1018  4028  ;               not being used currently.
                           1018  4029  ;
                           1018  4030
                           1018  4031
            57    53   D0  1018  4032          MOVL    R3,R7                     ; Set up variable ptr to dummy buffer
                           101B  4033
            87    85   D0  101B  4034          MOVL    (R5)+,(R7)+               ; Get "CPU busy" from SYSTEM
         87    10 A4   7D  101E  4035          MOVQ    16(R4),(R7)+              ; Get 8 items from STATES
         87    18 A4   7D  1022  4036          MOVQ    24(R4),(R7)+
         87    2C A4   7D  1026  4037          MOVQ    44(R4),(R7)+
         87    0C A4   D0  102A  4038          MOVL    12(R4),(R7)+
         87    04 A4   D0  102E  4039          MOVL    4(R4),(R7)+
                           1032  4040
   50   FFFFFFF7'8F    D0  1032  4041          MOVL    #<ECOUNT_SYS_SINGLE-9>,R0 ; No. of items to get from SYSTEM
                           1039  4042  10$:
            87    85   D0  1039  4043          MOVL    (R5)+,(R7)+               ; Move a SYSTEM item into dummy buffer
               FA 50   F5  103C  4044          SOBGTR  R0,10$                    ; Loop to get all of them
                           103F  4045
                           103F  4046
```

```
                                 103F  ;
                                 103F  ; At this point,
                                 103F  ;
                                 103F  ;       R3 - points to the newly formed dummy statistics
                                 103F  ;       R6 - CDB ptr
                                 103F  ;       R8 - statistic code
                                 103F  ;       R11 - TM4 ptr
                                 103F  ;
                                 103F
     55   00000103'EF     DE     103F          MOVAL   FAOSTK,R5               ; Load address of display buffer
                                 1046
                                 1046  ;
                                 1046  ; Loop once for each element in this class. Pick up transformed value for
                                 1046  ; desired statistic from appropriate buffer within collection buffer
                                 1046  ; block. Do computation on transformed value if required, and place
                                 1046  ; whole and fractional portions of result into display buffer (FAOSTK).
                                 1046  ;
                                 1046
     6B   FFFFFFFF'8F     D0     1046          MOVL    #<ECOUNT_SYS_SINGLE-1>, - ; Set up number of elements to
                                 104D                  TM4$L_ECOUNT(R11)       ; ... display as a loop limit
                                 104D
     50   00000000'EF     DE     104D          MOVAL   ITMSTR_SYS_SINGLE,R0    ; Address of item-number string
                                 1054
                    52    D4     1054          CLRL    R2                      ; Clear element index register
                                 1056  FSS_BEG:
                54  80    9A     1056          MOVZBL  (R0)+,R4                ; Get next item number
                54  11    C4     1059          MULL2   #IDB$K_ILENGTH,R4       ; Compute index into IDB table
          54  0000'CF44   9E     105C          MOVAB   W^PERFTABLE[R4],R4      ; Address of IDB for this item
     00E0'CF    0A A4     B0     1062          MOVW    IDB$W_TYPE(R4),W^ITEM_TYPE ; Save item type for use below
                                 1068
                02  58    D1     1068          CMPL    R8,#AVE_STAT            ; AVERAGE statistic requested ?
                05  13           106B          BEQL    10$                     ; Br if yes
              FE83  30           106D          BSBW    INTORFL                 ; Process an integer or floating value
                03  11           1070          BRB     20$                     ; ... and go loop for next element
                                 1072  10$:
              FE11  30           1072          BSBW    INTAVE                  ; Process an integer average value
                                 1075  20$:
   FFDB 52    01    6B    F1     1075          ACBL    TM4$L_ECOUNT(R11),#1,R2,FSS_BEG ; Loop once for each element
                                 107B
                    56    DD     107B          PUSHL   R6                      ; Pass CDB address
                    55    DD     107D          PUSHL   R5                      ; Pass current FAOSTK address
    00001089'EF     02    FB     107F          CALLS   #2,FDB_SYS_TOP          ; Compute and stack the four tops
                                 1086
              FDE2  31           1086          BRW     FDB_RET                 ; Go return
                                 1089
```

```
      1089          .SBTTL  FDB_SYS_TOP - Process TOPs for SYSTEM class
      1089
      1089 ;++
      1089 ;
      1089 ; FUNCTIONAL DESCRIPTION:
      1089 ;
      1089 ;          TBS
      1089 ;
      1089 ; INPUTS:
      1089 ;
      1089 ;
      1089 ; OUTPUTS:
      1089 ;
      1089 ;
      1089 ; IMPLICIT OUTPUTS:
      1089 ;
      1089 ;
      1089 ; ROUTINE VALUE:
      1089 ;
      1089 ;          R0 = SS$_NORMAL
      1089 ;
      1089 ; SIDE EFFECTS:
      1089 ;
      1089 ;          none
      1089 ;
      1089 ;--
      1089
0FFC  1089 .ENTRY   FDB_SYS_TOP, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
```

```
                                     108B  4123
          57   00000000'8F      D0   108B  4124              MOVL    #<PROCS_CLSNO*CDB$K_SIZE>,R7 ; Compute offset to PROCESSES CDB
        50   00000000'EF47      9E   1092  4125              MOVAB   CDBHEAD[R7],R0             ; Index to CDB address
                                     109A  4126                                                 ; NOTE -- TOP_DIFFS needs R0 set up this way
               57    2E A0      D0   109A  4127              MOVL    CDB$A_BUFFERS(R0),R7       ; Load buffer block ptr
                  57    67      D0   109E  4128              MOVL    MBP$A_BUFFERA(R7),R7       ; Load collection buffer ptr
                                     10A1  4129
                                     10A1  4130 ;
                                     10A1  4131 ; If not first TOP display event, calculate S_TOP_TICKS (the
                                     10A1  4132 ; number of clock ticks (10ms units) since previous entry
                                     10A1  4133 ; to FDB_SYS_TOP.
                                     10A1  4134 ;
                                     10A1  4135
          52   00000000'EF      D0   10A1  4136              MOVL    MCAPTR,R2                  ; Get MCA pointer
            19 32 A2    09      E1   10A8  4137              BBC     #MCA$V_S_TOP_DISP,MCA$W_FLAGS(R2),5$
                                     10AD  4138                                                 ; Skip TOP_TICKS calc if 1st time thru
              52     03 A7      7D   10AD  4139              MOVQ    MNR_CLS$Q_STAMP(R7),R2     ; Current system time to temp regs
             52    002F'CF      C2   10B1  4140              SUBL2   W^S_TOP_TIME,R2            ; Calc low-order in sys units
             53    0033'CF      D9   10B6  4141              SBWC    W^S_TOP_TIME+4,R3          ; Calc high-order in sys units
 0037'CF   52    000186A0 8F    7B   10BB  4142              EDIV    #100000,R2,W^S_TOP_TICKS,R2 ; Calc interval ticks (10ms units)
                        52           10C5
                                     10C6  4143                                                 ; ... for use below
                                     10C6  4144 5$:
        002F'CF      03 A7      7D   10C6  4145              MOVQ    MNR_CLS$Q_STAMP(R7),W^S_TOP_TIME ; Save curr time for next disp even
                                     10CC  4146
                                     10CC  4147 ;
                                     10CC  4148 ; Set up registers to call TOP_DIFFS routine to get the DIFF array
                                     10CC  4149 ; filled in, which contains the incremental differences in the
                                     10CC  4150 ; requested resource since previous display event. R0 and R7
                                     10CC  4151 ; already set up.
                                     10CC  4152 ;
                                     10CC  4153
          54   0000003B'EF      DE   10CC  4154              MOVAL   SYS_TOP_VEC,R4             ; Get ptr to vector of top buffers
                  52    84      D0   10D3  4155              MOVL    (R4)+,R2                   ; Get DATA array ptr
                  53    84      D0   10D6  4156              MOVL    (R4)+,R3                   ; Get DIFF array ptr
                  55    84      D0   10D9  4157              MOVL    (R4)+,R5                   ; Get PID array ptr
                  56    84      D0   10DC  4158              MOVL    (R4)+,R6                   ; Get ADDR array ptr
                  5B    2B      D0   10DF  4159              MOVL    #MNR_PRO$L_CPUTIM,R11      ; Get CPUTIM offset
                0081    8F      BB   10E2  4160              PUSHR   #^M<R0,R7>                 ; Save since changed by TOP_DIFFS
                      0304      30   10E6  4161              BSBW    TOP_DIFFS                  ; Calc diffs over the last interval
                0081    8F      BA   10E9  4162              POPR    #^M<R0,R7>                 ; Restore
                                     10ED  4163
                  52    84      D0   10ED  4164              MOVL    (R4)+,R2                   ; Get DATA array ptr
                  53    84      D0   10F0  4165              MOVL    (R4)+,R3                   ; Get DIFF array ptr
                  55    84      D0   10F3  4166              MOVL    (R4)+,R5                   ; Get PID array ptr
                  56    84      D0   10F6  4167              MOVL    (R4)+,R6                   ; Get ADDR array ptr
                  5B    27      D0   10F9  4168              MOVL    #MNR_PRO$L_PAGEFLTS,R11    ; Get PAGEFLTS offset
                0081    8F      BB   10FC  4169              PUSHR   #^M<R0,R7>                 ; Save since changed by TOP_DIFFS
                      02EA      30   1100  4170              BSBW    TOP_DIFFS                  ; Calc diffs over the last interval
                0081    8F      BA   1103  4171              POPR    #^M<R0,R7>                 ; Restore
                                     1107  4172
                  52    84      D0   1107  4173              MOVL    (R4)+,R2                   ; Get DATA array ptr
                  53    84      D0   110A  4174              MOVL    (R4)+,R3                   ; Get DIFF array ptr
                  55    84      D0   110D  4175              MOVL    (R4)+,R5                   ; Get PID array ptr
                  56    84      D0   1110  4176              MOVL    (R4)+,R6                   ; Get ADDR array ptr
                  5B    23      D0   1113  4177              MOVL    #MNR_PRO$L_DIOCNT,R11      ; Get DIOCNT offset
                0081    8F      BB   1116  4178              PUSHR   #^M<R0,R7>                 ; Save since changed by TOP_DIFFS
              02D0   30  111A          BSBW    TOP_DIFFS                  ; Calc diffs over the last interval
           0081 8F   BA  111D          POPR    #^M<R0,R7>                 ; Restore
                         1121
            52  84   D0  1121          MOVL    (R4)+,R2                   ; Get DATA array ptr
            53  84   D0  1124          MOVL    (R4)+,R3                   ; Get DIFF array ptr
            55  84   D0  1127          MOVL    (R4)+,R5                   ; Get PID array ptr
            56  84   D0  112A          MOVL    (R4)+,R6                   ; Get ADDR array ptr
            5B  2F   D0  112D          MOVL    #MNR_PRO$L_BIOCNT,R11      ; Get BIOCNT offset
           0081 8F   BB  1130          PUSHR   #^M<R0,R7>                 ; Save since changed by TOP_DIFFS
              02B6   30  1134          BSBW    TOP_DIFFS                  ; Calc diffs over the last interval
           0081 8F   BA  1137          POPR    #^M<R0,R7>                 ; Restore
                         113B  ;
                         113B  ; Now all DIFF arrays are established. Loop through each one to find
                         113B  ; the top user. Note: TOP_DIFFS places MAXPROCESSCNT in R11.
                         113B  ;
                         113B
                         113B
        55    04 AC  D0  113B          MOVL    4(AP),R5                   ; Get current position in FAOSTK
        56    08 AC  D0  113F          MOVL    8(AP),R6                   ; Get CDB address for later use
   54  0000003B'EF   DE  1143          MOVAL   SYS_TOP_VEC,R4             ; Get addr of TOP arrays
   51  00000000'EF   D0  114A          MOVL    MCAPTR,R1                  ; Get pointer to MCA
      0A 32 A1   09  E2  1151          BBSS    #MCA$V_S_TOP_DISP,MCA$W_FLAGS(R1),10$ ; Br if not first disp
                         1156                                                      ; ... and always set bit
                40   10  1156          BSBB    CLEAR_STACK                ; If first display event, simply
                3E   10  1158          BSBB    CLEAR_STACK                ; ... stack zeroes
                3C   10  115A          BSBB    CLEAR_STACK                ; ...
                3A   10  115C          BSBB    CLEAR_STACK                ; ...
                         115E
                37   11  115E          BRB     20$                        ; ... and go return
                         1160  10$:
        51    04 A4  D0  1160          MOVL    4(R4),R1                   ; Get addr of CPUTIM DIFF array
        53    0C A4  D0  1164          MOVL    12(R4),R3                  ; ... and ADDR array
                52   D4  1168          CLRL    R2                         ; ... and set up index for CALC_BAR
              0054   30  116A          BSBW    STACK_TOP                  ; Find the top and stack its info
        51    14 A4  D0  116D          MOVL    20(R4),R1                  ; Get addr of PAGEFLTS DIFF array
        53    1C A4  D0  1171          MOVL    28(R4),R3                  ; ... and ADDR array
           52   0B   D0  1175          MOVL    #11,R2                     ; ... and set up index for CALC_BAR
              0046   30  1178          BSBW    STACK_TOP                  ; Find the top and stack its info
        51    24 A4  D0  117B          MOVL    36(R4),R1                  ; Get addr of DIRIO DIFF array
        53    2C A4  D0  117F          MOVL    44(R4),R3                  ; ... and ADDR array
           52   0F   D0  1183          MOVL    #15,R2                     ; ... and set up index for CALC_BAR
              0038   30  1186          BSBW    STACK_TOP                  ; Find the top and stack its info
        51    34 A4  D0  1189          MOVL    52(R4),R1                  ; Get addr of BUFIO DIFF array
        53    3C A4  D0  118D          MOVL    60(R4),R3                  ; ... and ADDR array
           52   10   D0  1191          MOVL    #16,R2                     ; ... and set up index for CALC_BAR
              002A   30  1194          BSBW    STACK_TOP                  ; Find the top and stack its info
                         1197  20$:
                     04  1197          RET                                ; Return
                         1198
                         1198  CLEAR_STACK:
                85   7C  1198          CLRQ    (R5)+                      ; Stack null process name
                85   7C  119A          CLRQ    (R5)+                      ; ....
        85    F0 A5  9A  119C          MOVZBL  -16(R5),(R5)+              ; Length of process name to FAO stack
     85   55    13   C3  11A0          SUBL3   #19,R5,(R5)+               ; Address of process name to FAO stack
                85   7C  11A4          CLRQ    (R5)+                      ; Stack dummy value and bar value
                     05  11A6          RSB
                         11A7
```

```
                        11A7  FIND_TOP:
                50  D4  11A7          CLRL    R0                          ; Init index
        5A    6140  D0  11A9          MOVL    (R1)[R0],R10                ; Get first value
                58  D4  11AD          CLRL    R8                          ; ... and first index
                        11AF  10$:
      6140      5A  D1  11AF          CMPL    R10,(R1)[R0]                ; This item greater than current best?
                07  18  11B3          BGEQ    20$                         ; Br if no
        5A    6140  D0  11B5          MOVL    (R1)[R0],R10                ; Found a new best
           58   50  D0  11B9          MOVL    R0,R8                       ; Save its index
                        11BC  20$:
     EF 50      5B  F2  11BC          AOBLSS  R11,R0,10$                  ; Loop MAXPROCESSCNT times
                    05  11C0          RSB                                 ; Return
                        11C1
                        11C1  STACK_TOP:
                        11C1
                E4  10  11C1          BSBB    FIND_TOP                    ; Find the top process
                        11C3
                        11C3  ;
                        11C3  ; At this point, R10 has the top value, and R8 has its index value.
                        11C3  ;
                        11C3
     50      08 A4  D0  11C3          MOVL    8(R4),R0                    ; Get addr of CPUTIM PID array
     50       6048  D0  11C7          MOVL    (R0)[R8],R0                 ; Get PID for top process
                50  B5  11CB          TSTW    R0                          ; Is it the NULL process?
                05  12  11CD          BNEQ    10$                         ; Br if no
              6148  D4  11CF          CLRL    (R1)[R8]                    ; NULL process -- zero its DIFF val
                D3  10  11D2          BSBB    FIND_TOP                    ; ... and go find another top
                        11D4  10$:
                5A  D5  11D4          TSTL    R10                         ; Top value zero ?
                04  12  11D6          BNEQ    30$                         ; Br if no
                BE  10  11D8          BSBB    CLEAR_STACK                 ; Yes, simply stack zeroes
                2C  11  11DA          BRB     40$                         ; ... and go exit
                        11DC  30$:
     58       6348  D0  11DC          MOVL    (R3)[R8],R8                 ; Get addr of process data block
     85      0B A8  7D  11E0          MOVQ    MNR_PRO$O_LNAME(R8),(R5)+ ; Process name cstring to FAO stack
     85      13 A8  7D  11E4          MOVQ    MNR_PRO$O_LNAME+8(R8),(R5)+ ; .....
     85      F0 A5  9A  11E8          MOVZBL  -16(R5),(R5)+               ; Length of process name to FAO stack
  85    55      13  C3  11EC          SUBL3   #19,R5,(R5)+                ; Address of process name to FAO stack
                        11F0
                        11F0  ;
                        11F0  ; Transform the DIFF value (in R10) from a delta to a
                        11F0  ; rate/second. Then place it into the FAOSTK and ship
                        11F0  ; it off to the CALC_BAR subroutine to insert the
                        11F0  ; bar character count into FAOSTK.
                        11F0  ;
                        11F0
        5A      5A  4E  11F0          CVTLF   R10,R10                     ; Get floating value over interval
   50     0037'CF   4E  11F3          CVTLF   W^S_TOP_TICKS,R0            ; Get floating ticks over interval
50   000043C8 8F    46  11F8          DIVF2   #100,R0                     ; Get floating seconds over interval
        5A      50  46  11FF          DIVF2   R0,R10                      ; Get floating rate per second
        85      5A  4A  1202          CVTFL   R10,(R5)+                   ; Move longword rate/sec to FAO stack
              FD26  30  1205          BSBW    CALC_BAR                    ; Calculate and stack bar chars required
                        1208                                              ; CALC_BAR destroys regs R7 and R9, and
                        1208                                              ; ... updates R5 to point to the next
                        1208                                              ; ... available longword in the FAO stack
                        1208  40$:
                    05  1208          RSB                                 ; Return
```

```
1209  4293
```

```
                                     1209  ;
                                     1209  ; FDB_SYS_ALL
                                     1209  ;
                                     1209  ; Fill a dummy MBP and 4 dummy statistics buffers (STATS, MIN, MAX, and SUM).
                                     1209  ; The dummy statistics buffers will form a hybrid between the MODES class
                                     1209  ; and the SYSTEM class.
                                     1209  ;
                                     1209  ; Inputs:
                                     1209  ;
                                     1209  ;       R0 - R5 scratch
                                     1209  ;       R6  - address of SYSTEM CDB.
                                     1209  ;       R10 - address of dummy MBP followed by the 4 statistics buffers.
                                     1209  ;       R11 - address of TM4, a temporary stack area
                                     1209  ;
                                     1209  ; Implicit Outputs:
                                     1209  ;
                                     1209  ;       The dummy MBP and buffers are filled.
                                     1209  ;       TM4$L_ECOUNT contains number of elements to display.
                                     1209  ;       TM4$A_ITMSTR points to the display item string.
                                     1209  ;
                                     1209
                                     1209 FDB_SYS_ALL:
                                     1209
          50   00000000'8F    D0     1209         MOVL    #<CDB$K_SIZE*MODES_CLSNO>,R0 ; Compute offset to MODES CDB
        50   00000000'EF40    9E     1210         MOVAB   CDBHEAD[R0],R0          ; ... get its CDB address
                   51   2E A0 D0     1218         MOVL    CDB$A_BUFFERS(R0),R1    ; ... and MBP ptr for later use
                   52   2E A6 D0     121C         MOVL    CDB$A_BUFFERS(R6),R2    ; Get same for SYSTEM class
                                     1220
              53   5A   28    C1     1220         ADDL3   #MBP$K_SIZE,R10,R3      ; Compute address of dummy STATS buffer
                                     1224
                                     1224  ;
                                     1224  ; Fill STATS buffer with ECOUNT_SYS_ALL longwords
                                     1224  ; starting at address in R3.
                                     1224  ;
                                     1224
                   08 AA   53 D0     1224         MOVL    R3,MBP$A_STATS(R10)     ; Load addr of dummy STATS buffer
                   54   08 A1 D0     1228         MOVL    MBP$A_STATS(R1),R4      ; Get MODES STATS buffer ptr
                        55 07 D0     122C         MOVL    #7,R5                   ; ... and no. of items to pick up
                           6F 10     122F         BSBB    MOVE_ITEMS              ; Move the items into the dummy buffer
                                     1231
              54   08 A2   08 C1     1231         ADDL3   #8,MBP$A_STATS(R2),R4   ; Get SYSTEM STATS buff ptr (skips 2 items)
          55   FFFFFFF9'8F    D0     1236         MOVL    #<ECOUNT_SYS_ALL-7>,R5  ; ... and no. of items to pick up
                           61 10     123D         BSBB    MOVE_ITEMS              ; Move the items into the dummy buffer
                                     123F
                                     123F  ;
                                     123F  ; Fill MIN buffer with ECOUNT_SYS_ALL longwords
                                     123F  ; starting at address in R3.
                                     123F  ;
                                     123F
                   0C AA   53 D0     123F         MOVL    R3,MBP$A_MIN(R10)       ; Load addr of dummy MIN buffer
                   54   0C A1 D0     1243         MOVL    MBP$A_MIN(R1),R4        ; Get MODES MIN buffer ptr
                        55 07 D0     1247         MOVL    #7,R5                   ; ... and no. of items to pick up
                           54 10     124A         BSBB    MOVE_ITEMS              ; Move the items into the dummy buffer
                                     124C
              54   0C A2   08 C1     124C         ADDL3   #8,MBP$A_MIN(R2),R4     ; Get SYSTEM MIN buff ptr (skips 2 itemS)
          55   FFFFFFF9'8F    D0     1251         MOVL    #<ECOUNT_SYS_ALL-7>,R5  ; ... and no. of items to pick up
                           46 10     1258         BSBB    MOVE_ITEMS              ; Move the items into the dummy buffer
```

```
                                  125A  4352
                                  125A  4353 ;
                                  125A  4354 ; Fill MAX buffer with ECOUNT_SYS_ALL longwords
                                  125A  4355 ; starting at address in R3.
                                  125A  4356 ;
                                  125A  4357
              10 AA    53   D0    125A  4358         MOVL    R3,MBP$A_MAX(R10)        ; Load addr of dummy MAX buffer
           54    10 A1      D0    125E  4359         MOVL    MBP$A_MAX(R1),R4         ; Get MODES MAX buffer ptr
                 55    07   D0    1262  4360         MOVL    #7,R5                    ; ... and no. of items to pick up
                       39   10    1265  4361         BSBB    MOVE_ITEMS               ; Move the items into the dummy buffer
                                  1267  4362
      54    10 A2      08   C1    1267  4363         ADDL3   #8,MBP$A_MAX(R2),R4      ; Get SYSTEM MAX buff ptr (skips 2 itemS)
      55    FFFFFFF9'8F     D0    126C  4364         MOVL    #<ECOUNT_SYS_ALL-7>,R5   ; ... and no. of items to pick up
                       2B   10    1273  4365         BSBB    MOVE_ITEMS               ; Move the items into the dummy buffer
                                  1275  4366
                                  1275  4367 ;
                                  1275  4368 ; Fill SUM buffer with ECOUNT_SYS_ALL longwords
                                  1275  4369 ; starting at address in R3.
                                  1275  4370 ;
                                  1275  4371
              14 AA    53   D0    1275  4372         MOVL    R3,MBP$A_SUM(R10)        ; Load addr of dummy SUM buffer
           54    14 A1      D0    1279  4373         MOVL    MBP$A_SUM(R1),R4         ; Get MODES SUM buffer ptr
                 55    07   D0    127D  4374         MOVL    #7,R5                    ; ... and no. of items to pick up
                       1E   10    1280  4375         BSBB    MOVE_ITEMS               ; Move the items into the dummy buffer
                                  1282  4376
      54    14 A2      08   C1    1282  4377         ADDL3   #8,MBP$A_SUM(R2),R4      ; Get SYSTEM SUM buff ptr (skips 2 itemS)
      55    FFFFFFF9'8F     D0    1287  4378         MOVL    #<ECOUNT_SYS_ALL-7>,R5   ; ... and no. of items to pick up
                       10   10    128E  4379         BSBB    MOVE_ITEMS               ; Move the items into the dummy buffer
                                  1290  4380
                                  1290  4381
      6B    00000000'8F     D0    1290  4382         MOVL    #ECOUNT_SYS_ALL, -       ; Load number of elements to display
                                  1297  4383                 TM4$L_ECOUNT(R11)
   04 AB    00000000'EF     DE    1297  4384         MOVAL   ITMSTR_SYS_ALL, -
                                  129F  4385                 TM4$A_ITMSTR(R11)        ; ... and addr of display item string
                                  129F  4386
                            05    129F  4387         RSB                              ; Return to caller
                                  12A0  4388
                                  12A0  4389
                                  12A0  4390 ;
                                  12A0  4391 ; MOVE_ITEMS moves a consecutive number of longwords (number in R5)
                                  12A0  4392 ; from location R4 to location R3. R4 and R3 are auto-incremented.
                                  12A0  4393 ;
                                  12A0  4394
                                  12A0  4395 MOVE_ITEMS:
                                  12A0  4396 10$:
                 83    84   D0    12A0  4397         MOVL    (R4)+,(R3)+              ; Move an item into dummy buffer
                    FA 55   F5    12A3  4398         SOBGTR  R5,10$                   ; Loop to get all of them
                            05    12A6  4399         RSB
                                  12A7  4400
```

```
12A7          .SBTTL  FILL_TOP - Fill Display Buffer for TOP PROCESSES
12A7
12A7    ;++
12A7    ;
12A7    ; FUNCTIONAL DESCRIPTION:
12A7    ;
12A7    ;       Calculates the TOP 8 PROCESSES since the last display
12A7    ;       event, and fills the display buffer (FAOSTK) with data
12A7    ;       for later display.
12A7    ;
12A7    ; INPUTS:
12A7    ;
12A7    ;        4(AP) - CDB (Class Descriptor Block) pointer
12A7    ;                for the PROCESSES class.
12A7    ;
12A7    ; IMPLICIT INPUTS:
12A7    ;
12A7    ;       FAOSTK - FAO parameter list for a TOP screen
12A7    ;       MCAPTR - Pointer to MCA (Monitor Communication Area)
12A7    ;
12A7    ; OUTPUTS:
12A7    ;
12A7    ;       None
12A7    ;
12A7    ; IMPLICIT OUTPUTS:
12A7    ;
12A7    ;       Entire display buffer (FAOSTK) filled with data for
12A7    ;       eventual display.
12A7    ;
12A7    ;       TOP_PROCS byte filled with number of TOP processes
12A7    ;       (with non-zero values) to display.
12A7    ;
12A7    ; ROUTINE VALUE:
12A7    ;
12A7    ;       R0 = SS$_NORMAL
12A7    ;
12A7    ; SIDE EFFECTS:
12A7    ;
12A7    ;       none
12A7    ;
12A7    ; REGISTER USAGE:
12A7    ;
12A7    ;       R0  = Scratch
12A7    ;       R1  = Scratch, addr of MBP
12A7    ;       R2  = Addr of DATA array
12A7    ;       R3  = Addr of DIFF array
12A7    ;       R4  = Addr of ORDER array
12A7    ;       R5  = Addr of PID array
12A7    ;       R6  = Addr of ADDR array
12A7    ;       R7  = Pointer to collection buffer data block
12A7    ;       R8  = Process index (PIX) for current data block
12A7    ;       R9  = Current process index (from 0 to MNR_SYI$W_MAXPRCCT)
12A7    ;       R10 = Number of data blocks (processes) in coll buff
12A7    ;       R11 = Pointer to monitored data item in coll buff data block;
12A7    ;             Also, Max process count (from MNR_SYI$W_MAXPRCCT)
12A7    ;--
12A7
```

```
OFFC  12A7  4459 .ENTRY  FILL_TOP, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
```

```
                 51     04 AC    D0  12A9                MOVL    4(AP),R1                  ; Load CDB pointer
                 5B     42 A1    9A  12AD                MOVZBL  CDB$B_ST(R1),R11          ; Get PROCESSES display type code
                                     12B1                CASE    R11,<REG,TOPC,TOPD,TOPB,TOPF>,W ; Go set offset based on type
                           00    11  12BF                BRB     TOPC                      ; If out of range, do a TOPCPU
                                     12C1
                                     12C1      REG:                                        ; Regular PROCESSES display (should not get)
                                     12C1      TOPC:                                       ; TOPCPU display
                      5B   2B    D0  12C1                MOVL    #MNR_PRO$L_CPUTIM,R11     ; Get offset into PROCESSES data block
                           0D    11  12C4                BRB     FT_CASE                   ; Join common code
                                     12C6      TOPD:                                       ; TOPDIO display
                      5B   23    D0  12C6                MOVL    #MNR_PRO$L_DIOCNT,R11     ; Get offset into PROCESSES data block
                           08    11  12C9                BRB     FT_CASE                   ; Join common code
                                     12CB      TOPB:                                       ; TOPBIO display
                      5B   2F    D0  12CB                MOVL    #MNR_PRO$L_BIOCNT,R11     ; Get offset into PROCESSES data block
                           03    11  12CE                BRB     FT_CASE                   ; Join common code
                                     12D0      TOPF:                                       ; TOPFAULT display
                      5B   27    D0  12D0                MOVL    #MNR_PRO$L_PAGEFLTS,R11   ; Get offset into PROCESSES data block
                                     12D3
                                     12D3      FT_CASE:                                    ; Common CASE return
                                     12D3
                 51     2E A1    D0  12D3                MOVL    CDB$A_BUFFERS(R1),R1      ; Load buffer block ptr
                    57     61    D0  12D7                MOVL    MBP$A_BUFFERA(R1),R7      ; Load collection buffer ptr
                                     12DA
                                     12DA      ;
                                     12DA      ; If not first TOP display event, calculate TOP_TICKS (the
                                     12DA      ; number of clock ticks (10ms units) since previous entry
                                     12DA      ; to FILL_TOP.
                                     12DA      ;
                                     12DA
          52   00000000'EF       D0  12DA                MOVL    MCAPTR,R2                 ; Get MCA pointer
            19 32 A2       07    E1  12E1                BBC     #MCA$V_TOP_DISP,MCA$W_FLAGS(R2),5$
                                     12E6                                                  ; Skip TOP_TICKS calc if 1st time thru
                 52     03 A7    7D  12E6                MOVQ    MNR_CLS$Q_STAMP(R7),R2    ; Current system time to temp regs
               52     0023'CF    C2  12EA                SUBL2   W^TOP_TIME,R2             ; Calc low-order in sys units
               53     0027'CF    D9  12EF                SBWC    W^TOP_TIME+4,R3           ; Calc high-order in sys units
 002B'CF  52   000186A0 8F       7B  12F4                EDIV    #100000,R2,W^TOP_TICKS,R2 ; Calc interval ticks (10ms units)
                           52        12FE
                                     12FF                                                  ; ... for use in MOVE_TOP8 rtn below
                                     12FF      5$:
         0023'CF    03 A7        7D  12FF                MOVQ    MNR_CLS$Q_STAMP(R7),W^TOP_TIME ; Save curr time for next disp event
                                     1305
                                     1305      ;
                                     1305      ; Set up array pointers in preparation for calculations of difference
                                     1305      ; values for each process for the monitored item over the last interval.
                                     1305      ;
                                     1305
                 52     08 A1    D0  1305                MOVL    MBP$A_DATA(R1),R2         ; Load DATA array ptr
                 53     0C A1    D0  1309                MOVL    MBP$A_DIFF(R1),R3         ; Load DIFF array ptr
                 54     10 A1    D0  130D                MOVL    MBP$A_ORDER(R1),R4        ; Load ORDER array ptr
                 55     14 A1    D0  1311                MOVL    MBP$A_PID(R1),R5          ; Load PID array ptr
                 56     18 A1    D0  1315                MOVL    MBP$A_ADDR(R1),R6         ; Load ADDR array ptr
                 50     04 AC    D0  1319                MOVL    4(AP),R0                  ; Load CDB pointer
                                     131D
                        00CD     30  131D                BSBW    TOP_DIFFS                 ; Calculate the diffs over last int
                                     1320
          51   00000000'EF       D0  1320                MOVL    MCAPTR,R1                 ; Get pointer to MCA
            0E 32 A1       07    E3  1327                BBCS    #MCA$V_TOP_DISP,MCA$W_FLAGS(R1),80$
```

```
                         132C  4517                                          ; If first top display event, don't
                         132C  4518                                          ; ... sort or move (and set bit for future)
          0800 8F    BB  132C  4519         PUSHR   #^M<R11>                 ; Save max process count
                         1330  4520
               14    10  1330  4521         BSBB    SORT_PROCS               ; Sort the top 8 processes
                         1332  4522
          0800 8F    BA  1332  4523         POPR    #^M<R11>                 ; Restore max process count
                         1336  4524
               4B    10  1336  4525         BSBB    MOVE_TOP8                ; Insert data for TOP 8 into FAOSTK
                         1338  4526
               04    11  1338  4527         BRB     90$                      ; Go return
                         133A  4528 80$:
          0022'CF    94  133A  4529         CLRB    W^TOP_PROCS              ; No procs to display on 1st time
                         133E  4530 90$:
  50   00000000'EF   D0  133E  4531         MOVL    NORMAL,R0                ; Indicate normal status
                     04  1345  4532         RET                              ; Return
```

```
                                  1346  ;
                                  1346  ; SORT_PROCS
                                  1346  ;
                                  1346  ; Set up the ORDER array to contain the processes indices.
                                  1346  ;
                                  1346  ; R11 contains max process count.
                                  1346  ;
                                  1346  ; This subroutine destroys registers R0,R1,R7,R8,R9,R10,R11.
                                  1346  ;
                                  1346
                                  1346  SORT_PROCS:
                                  1346
                          51   D4 1346          CLRL    R1                      ; Zero first process index
               6441       51   D0 1348  10$:    MOVL    R1,(R4)[R1]             ; Load index into corresponding ORDER elem
              F8 51       5B   F2 134C          AOBLSS  R11,R1,10$              ; Do all elements of ORDER array
                                  1350
                                  1350  ;
                                  1350  ; Go through the DIFF array and re-position elements in the
                                  1350  ; ORDER array using a bubble sort. When the following two-level
                                  1350  ; loop is complete, the highest-numbered 8 elements of the ORDER
                                  1350  ; array will contain the process index numbers of the TOP 8
                                  1350  ; consumers of the monitored resource.
                                  1350  ;
                                  1350
                          5B   D7 1350          DECL    R11                     ; Get highest process index (PIX)
                 5A   5B  07   C3 1352          SUBL3   #7,R11,R10              ; Get 8th from the highest PIX
                                  1356  20$:
                      59  01   D0 1356          MOVL    #1,R9                   ; Init loop index of inner loop
                                  1359  30$:
                 51   59  01   C3 1359          SUBL3   #1,R9,R1                ; R1 is always one less than R9
                  57    6449   D0 135D          MOVL    (R4)[R9],R7             ; Get PIX from current ORDER element
                  58    6441   D0 1361          MOVL    (R4)[R1],R8             ; Get PIX from previous ORDER element
               6348     6347   D1 1365          CMPL    (R3)[R7],(R3)[R8]       ; Compare curr DIFF val with previous
                          08   18 136A          BGEQ    40$                     ; Curr is not less -- no switching
               6449       58   D0 136C          MOVL    R8,(R4)[R9]             ; Curr is less -- switch PIX in current
               6441       57   D0 1370          MOVL    R7,(R4)[R1]             ; ... ORDER elt with that in prev ORDER elt
                                  1374  40$:
              E1 59       5B   F3 1374          AOBLEQ  R11,R9,30$              ; Loop through all elements of ORDER
                                  1378                                          ; ... array except the ones on the high
                                  1378                                          ; ... end which already have TOP values
                                  1378
FFD4 5B   FFFFFFFF 8F     5A   F1 1378          ACBL    R10,#-1,R11,20$         ; Loop 8 times to "bubble down" PIX's
                                  1382                                          ; ... for the 8 largest consumers
                                  1382
                               05 1382          RSB                             ; Return
                                  1383
                          1383 ;
                          1383 ; MOVE_TOP8
                          1383 ;
                          1383 ; Move data for the top 8 (or fewer) processes
                          1383 ; into FAOSTK for later display.
                          1383 ;
                          1383 ; R11 contains max process count.
                          1383 ;
                          1383 ; This subroutine destroys registers R0,R1,R5,R7,R8,R9,R10,R11.
                          1383 ;
                          1383
                          1383 MOVE_TOP8:
                          1383
      55   0103'CF    DE  1383                 MOVAL   W^FAOSTK,R5                ; Get pointer to display buffer
         51     01    D0  1388                 MOVL    #1,R1                      ; Init count of procs that have
                          138B                                                    ; ... DIFF value > 0
                          138B 10$:                                               ; Beginning of TOP 8 loop
                5B    D7  138B                 DECL    R11                        ; Point to next lower ORDER element
       58     644B    D0  138D                 MOVL    (R4)[R11],R8               ; Get PIX from ORDER array
       5A     6348    D0  1391                 MOVL    (R3)[R8],R10               ; Get DIFF value for this TOP process
                4F    13  1395                 BEQL    40$                        ; Get out of loop if zero
       58     6648    D0  1397                 MOVL    (R6)[R8],R8                ; Get ptr to process data block
                          139B                                                    ; ... from ADDR array
                          139B ;
                          139B ; NOTE -- at this point, R8 points to process data block
                          139B ; ... and R10 has DIFF value.
                          139B ;
                          139B
                          139B ;
                          139B ; Move individual items for this process from current data block
                          139B ; in collection buffer to longwords in FAO stack.
                          139B ;
                          139B
      50    04 AC     D0  139B                 MOVL    4(AP),R0                   ; Get PROCESSES CDB address
                          139F
      20 A0    33     B1  139F                 CMPW    #MNR_PRO$L_EPID,CDB$W_BLKLEN(R0) ; See if we have an EPID
               05     19  13A3                 BLSS    20$                        ; Br if we do
         85    68     D0  13A5                 MOVL    MNR_PRO$L_IPID(R8),(R5)+ ; Get the Internal PID
               04     11  13A8                 BRB     30$                        ; Go get process name
                          13AA 20$:
      85    33 A8     D0  13AA                 MOVL    MNR_PRO$L_EPID(R8),(R5)+ ; Get the Extended PID
                          13AE 30$:
      85    06 A8     7D  13AE                 MOVQ    MNR_PRO$O_LNAME(R8),(R5)+ ; Process name cstring to FAO stack
      85    13 A8     7D  13B2                 MOVQ    MNR_PRO$O_LNAME+8(R8),(R5)+ ; ....
      85    F0 A5     9A  13B6                 MOVZBL  -16(R5),(R5)+              ; Length of process name to FAO stack
   85    55    13     C3  13BA                 SUBL3   #19,R5,(R5)+               ; Address of process name to FAO stack
                          13BE
                          13BE ;
                          13BE ; Transform the DIFF value (in R10) from a delta to a
                          13BE ; rate/second. Then place it into the FAOSTK and ship
                          13BE ; it off to the CALC_BAR subroutine to insert the
                          13BE ; bar character count into FAOSTK.
                          13BE ;
                          13BE
         5A     5A    4E  13BE                 CVTLF   R10,R10                    ; Get floating value over interval
      50   002B'CF    4E  13C1                 CVTLF   W^TOP_TICKS,R0             ; Get floating ticks over interval
 50   000043C8 8F     46  13C6                 DIVF2   #100,R0                    ; Get floating seconds over interval
```

```
         5A   50   46  13CD  4638          DIVF2   R0,R10                   ; Get floating rate per second
         85   5A   4A  13D0  4639          CVTFL   R10,(R5)+                ; Move longword rate/sec to FAO stack
       0040   8F   BB  13D3  4640          PUSHR   #^M<R6>                  ; Save ADDR array pointer
    56   04   AC   D0  13D7  4641          MOVL    4(AP),R6                 ; ... and set up CDB ptr for CALC_BAR
           FB50    30  13DB  4642          BSBW    CALC_BAR                 ; Calculate and stack bar chars required
                       13DE  4643                                           ; CALC_BAR destroys regs R7 and R9, and
                       13DE  4644                                           ; ... updates R5 to point to the next
                       13DE  4645                                           ; ... available longword in the FAO stack
       0040   8F   BA  13DE  4646          POPR    #^M<R6>                  ; Restore ADDR array pointer
                       13E2  4647
      A5 51   08   F3  13E2  4648          AOBLEQ  #8,R1,10$                ; Get info from TOP 8 processes
                       13E6  4649 40$:
0022'CF  51   01   83  13E6  4650          SUBB3   #1,R1,W^TOP_PROCS        ; Adjust count of processes for
                       13EC  4651                                           ; ... DISPLAY_TOP routine
                       13EC  4652
                   05  13EC  4653          RSB                              ; Return
                       13ED  4654
```

```
                            13ED  TOP_DIFFS:
                            13ED
                            13ED  ;
                            13ED  ; Fill the DIFF array and ADDR array for the current collection buffer.
                            13ED  ;
                            13ED  ;
                            13ED  ; REGISTER INPUTS:
                            13ED  ;
                            13ED  ;       R0  = PROCESSES CDB ptr
                            13ED  ;       R1  = scratch
                            13ED  ;       R2  = DATA array ptr
                            13ED  ;       R3  = DIFF array ptr
                            13ED  ;       R5  = PID array ptr
                            13ED  ;       R6  = ADDR array ptr
                            13ED  ;       R7  = pointer to PROCESSES collection buffer
                            13ED  ;       R8-10 = scratch
                            13ED  ;       R11 = offset into PROCESSES data block for requested resource
                            13ED  ;
                            13ED
          57    0D     C0   13ED          ADDL2   #MNR_CLS$K_HSIZE,R7      ; Point to PROCESSES class prefix
       5A    04 A7     D0   13F0          MOVL    MNR_PRO$L_PCTINT(R7),R10 ; Get no. of procs in this coll buffer
          57    08     C0   13F4          ADDL2   #MNR_PRO$K_PSIZE,R7      ; Point to first data block
          5B    57     C0   13F7          ADDL2   R7,R11                   ; ... and to first monitored data item
                            13FA
  59   FFFFFFFF 8F     D0   13FA          MOVL    #-1,R9                   ; Init process index
```

```
                                1401  10$:
             58    67     3C    1401          MOVZWL  MNR_PRO$L_IPID(R7),R8    ; Get process index from next process
                                1404                                           ; ... in collection buffer
                                1404  20$:
                   59     D6    1404          INCL    R9                       ; Get next process index
             58    59     D1    1406          CMPL    R9,R8                    ; Any process slots not in coll buff?
                   05     18    1409          BGEQ    30$                      ; No -- go process this one
                 6349     D4    140B          CLRL    (R3)[R9]                 ; Yes -- clear DIFF array for this index
                   F4     11    140E          BRB     20$                      ; Loop back to check next index
                                1410  30$:
          67     6548     D1    1410          CMPL    (R5)[R8],MNR_PRO$L_IPID(R7) ; Same process as last time?
                   15     12    1414          BNEQU   40$                      ; No -- go zero out DIFF
                   6B     D5    1416          TSTL    (R11)                    ; Zero data item => swapped out
                   11     13    1418          BEQL    40$                      ; Swapped out -- go zero out DIFF
                 6248     D5    141A          TSTL    (R2)[R8]                 ; Swapped out last time?
                   0C     13    141D          BEQL    40$                      ; Yes -- go zero out DIFF
  6348     6B    6248     C3    141F          SUBL3   (R2)[R8],(R11),(R3)[R8]  ; Calculate DIFF
         6648      57     D0    1425          MOVL    R7,(R6)[R8]              ; Store proc data block ptr in ADDR array
                   07     11    1429          BRB     50$                      ; ... and continue
                                142B  40$:
                 6348     D4    142B          CLRL    (R3)[R8]                 ; Clear DIFF, indicating not a TOP candidate
         6548      67     D0    142E          MOVL    MNR_PRO$L_IPID(R7),(R5)[R8] ; Store PID
                                1432  50$:
         6248      6B     D0    1432          MOVL    (R11),(R2)[R8]           ; Store data item into DATA array
          51    20 A0     3C    1436          MOVZWL  CDB$W_BLKLEN(R0),R1      ; Get size of a data block
             57    51     C0    143A          ADDL2   R1,R7                    ; Point to next process in coll buffer
             5B    51     C0    143D          ADDL2   R1,R11                   ; ... and to next data item
             BE    5A     F5    1440          SOBGTR  R10,10$                  ; Loop once for each proc in coll buffer
                                1443
                                1443  ;
                                1443  ; A DIFF entry has been made for every index up through
                                1443  ; the last process in the collection buffer.
                                1443  ; The following loop clears the DIFF entry for each
                                1443  ; index between the last one already done and the
                                1443  ; last one in the DIFF array.
                                1443  ;
                                1443
                   58     D6    1443          INCL    R8                       ; Get next process index
  5B    00000000'EF       D0    1445          MOVL    SPTR,R11                 ; Get ptr to System Info Area
          5B    0B AB     3C    144C          MOVZWL  MNR_SYI$W_MAXPRCCT(R11),R11 ; Get max process count
             5B    58     D1    1450          CMPL    R8,R11                   ; Any more process slots?
                   07     18    1453          BGEQ    70$                      ; No -- skip clear loop
                                1455  60$:
                 6348     D4    1455          CLRL    (R3)[R8]                 ; Clear DIFF, indicating not a TOP candidate
                                1458
          F9 58    5B     F2    1458          AOBLSS  R11,R8,60$               ; Continue to end of DIFF array
                                145C  70$:
                                145C
                          05    145C          RSB                              ; Return to caller
```

```
145D          .SBTTL  SUMMARY_TOP - Set up Summary for TOP
145D
145D  ;++
145D  ;
145D  ; FUNCTIONAL DESCRIPTION:
145D  ;
145D  ;       This routine is called to do setup for Summary processing of
145D  ;       the SYSTEM or PROCESSES class with the TOP display format. It is not
145D  ;       called when summarizing with the regular PROCESSES display format
145D  ;       or the tabular SYSTEM display format.
145D  ;
145D  ;       The basic job of this routine is to call FDB_SYS_TOP or FILL_DISP_BUFF
145D  ;       with the first PROCESSES collection buffer of the MONITOR request.
145D  ;       This is accomplished by loading the pointer to the current collection
145D  ;       buffer with the first collection buffer pointer, and then doing
145D  ;       a normal FDB_SYS_TOP or FILL_DISP_BUFF call. The current collection
145D  ;       buffer pointer is then restored to its original value before returning
145D  ;       to caller.
145D  ;
145D  ; CALLING SEQUENCE:
145D  ;
145D  ;       CALLS #1,SUMMARY_TOP
145D  ;
145D  ; INPUTS:
145D  ;
145D  ;        4(AP) - address of a pointer to the CDB (Class Descriptor Block)
145D  ;                for either the SYSTEM or the PROCESSES class.
145D  ;
145D  ; IMPLICIT INPUTS:
145D  ;
145D  ;       MCAPTR - pointer to MCA (Monitor Communication Area)
145D  ;
145D  ;       MRBPTR - pointer to MRB (Monitor Request Block)
145D  ;
145D  ;       SPTR - pointer to SYI (System Information Area)
145D  ;
145D  ;       PROCS_CLSNO - class number for the PROCESSES class
145D  ;
145D  ;       SYS_TOP_VEC - vector of pointers to SYSTEM class TOP arrays
145D  ;
145D  ; OUTPUTS:
145D  ;
145D  ;       None.
145D  ;
145D  ; IMPLICIT OUTPUTS:
145D  ;
145D  ;       The 5 TOP arrays (DATA, DIFF, ORDER, PID, ADDR)
145D  ;       are filled with information from the first collection
145D  ;       buffer.
145D  ;
145D  ; ROUTINE VALUE:
145D  ;
145D  ;       R0 = SS$_NORMAL
145D  ;
145D  ; SIDE EFFECTS:
145D  ;
145D  ;       None
```

```
145D  4789 ;
145D  4790 ;--
```

```
                          0FFC  145D  4792 .ENTRY  SUMMARY_TOP,    ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                145F  4793
     51   00000000'EF   D0      145F  4794         MOVL    MCAPTR,R1                    ; Load MCA pointer
      00 32 A1    07    E5      1466  4795         BBCC    #MCA$V_TOP_DISP,MCA$W_FLAGS(R1),10$ ; Indicate no TOP displays
                                146B  4796 10$:                                         ; ... done yet for PROCESSES
      00 32 A1    09    E5      146B  4797         BBCC    #MCA$V_S_TOP_DISP,MCA$W_FLAGS(R1),20$ ; ... or SYSTEM class
                                1470  4798 20$:
       56   04 BC       D0      1470  4799         MOVL    @4(AP),R6                    ; Load CDB pointer
     51   00000000'EF   D0      1474  4800         MOVL    MRBPTR,R1                    ; Load MRB pointer
      43 43 A1    00    E1      147B  4801         BBC     #MRB$V_DISPLAY,MRB$W_FLAGS(R1),50$ ; No need to clear DATA array
                                1480  4802                                              ; ... if not already used for displaying
     52   00000000'EF   D0      1480  4803         MOVL    SPTR,R2                      ; Load SYI pointer
       52   0B A2       3C      1487  4804         MOVZWL  MNR_SYI$W_MAXPRCCT(R2),R2 ; Get max process count
      58   52    04     C5      148B  4805         MULL3   #4,R2,R8                     ; Pass size of DATA array
      10 4B A6    08    E0      148F  4806         BBS     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),30$ ; Br if SYSTEM class
                                1494  4807
                                1494  4808 ;
                                1494  4809 ; Call CLEAR_DATA to clear the DATA array for PROCESSES class
                                1494  4810 ;
                                1494  4811
       59   2E A6       D0      1494  4812         MOVL    CDB$A_BUFFERS(R6),R9         ; Pass address of DATA array
       59   08 A9       D0      1498  4813         MOVL    MBP$A_DATA(R9),R9            ; ...
        00000000'EF     16      149C  4814         JSB     CLEAR_DATA                   ; Clear DATA array in preparation for
                                14A2  4815                                              ; ... call to FILL_DISP_BUFF
                                14A2  4816                                              ; (CLEAR_DATA subrtn destroys R0-R5 and
                                14A2  4817                                              ; ... R8,R9)
                  1F    11      14A2  4818         BRB     50$                          ; Go continue.....
                                14A4  4819
                                14A4  4820 ;
                                14A4  4821 ; Call CLEAR_DATA to clear all 4 DATA arrays for SYSTEM class
                                14A4  4822 ;
                                14A4  4823
                                14A4  4824 30$:
     5A   0000003B'EF   DE      14A4  4825         MOVAL   SYS_TOP_VEC,R10              ; Get addr of vector of ptrs
            5B    58    D0      14AB  4826         MOVL    R8,R11                       ; Save array length
            57    04    D0      14AE  4827         MOVL    #4,R7                        ; Number of arrays to clear
                                14B1  4828 40$:
            59    6A    D0      14B1  4829         MOVL    (R10),R9                     ; R9 must contain array addr
            58    5B    D0      14B4  4830         MOVL    R11,R8                       ; R8 gets array length
        00000000'EF     16      14B7  4831         JSB     CLEAR_DATA                   ; Clear DATA array in preparation for
                                14BD  4832                                              ; ... call to FDB_SYS_TOP
                                14BD  4833                                              ; (CLEAR_DATA subrtn destroys R0-R5 and
                                14BD  4834                                              ; ... R8,R9)
            5A    10    C0      14BD  4835         ADDL2   #16,R10                      ; Point to next array
             EE 57      F5      14C0  4836         SOBGTR  R7,40$                       ; Loop back to process next one
```

```
                            14C3  4838 ;
                            14C3  4839 ; Load PROCESSES current collection buffer pointer (MBP$A_BUFFA)
                            14C3  4840 ; with first collection buffer pointer (MBP$A_BUFF1ST); then
                            14C3  4841 ; call FDB_SYS_TOP or FILL_DISP_BUFF to get the 5 TOP arrays
                            14C3  4842 ; loaded with data from the first collection buffer. Finally,
                            14C3  4843 ; restore original value of MBP$A_BUFFA for caller's use.
                            14C3  4844 ;
                            14C3  4845
                            14C3  4846 50$:
      50   00000000'8F   D0 14C3  4847         MOVL    #<PROCS_CLSNO*CDB$K_SIZE>,R0 ; Compute offset to PROCESSES CDB
    50   00000000'EF40   9E 14CA  4848         MOVAB   CDBHEAD[R0],R0             ; Index to CDB address
         52     2E A0    D0 14D2  4849         MOVL    CDB$A_BUFFERS(R0),R2       ; Load buffer block ptr
            58     62    D0 14D6  4850         MOVL    MBP$A_BUFFA(R2),R8         ; Save current buffer pointer
         62     04 A2    D0 14D9  4851         MOVL    MBP$A_BUFF1ST(R2),MBP$A_BUFFA(R2) ; Move first to current
                            14DD  4852
       1B 4B A6    08    E0 14DD  4853         BBS     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),60$ ; Br if SYSTEM class
                            14E2  4854
                            14E2  4855         ALLOC   4,R0,R1                    ; Allocate 12 (4+descr) stack bytes
                            14EF  4856                                            ; ... for FILL_DISP_BUFF call
            61     56    D0 14EF  4857         MOVL    R6,(R1)                    ; CDB pointer into allocated space
                   50    DD 14F2  4858         PUSHL   R0                         ; Push pointer to time quadword
                   51    DD 14F4  4859         PUSHL   R1                         ; Push address of CDB pointer
       F68C CF     02    FB 14F6  4860         CALLS   #2,FILL_DISP_BUFF          ; Fill 5 TOP arrays
                   0D    11 14FB  4861         BRB     70$                        ; Go continue.....
                            14FD  4862 60$:
                   56    DD 14FD  4863         PUSHL   R6                         ; Pass SYSTEM CDB address
       00000103'EF       DF 14FF  4864         PUSHAL  FAOSTK                     ; Pass display buffer address
       FB7F CF     02    FB 1505  4865         CALLS   #2,FDB_SYS_TOP             ; Fill 4 TOP arrays (doesn't use ORDER)
                            150A  4866 70$:
            62     58    D0 150A  4867         MOVL    R8,MBP$A_BUFFA(R2)         ; Restore current collection buffer ptr
                            150D  4868
            50     01    D0 150D  4869         MOVL    #SS$_NORMAL,R0             ; Indicate success
                         04 1510  4870         RET                                ; ... and return
```

```
                               1511         .SBTTL  DISPLAY_PROCS - Put PROCESSES Display Output to Screen
                               1511
                               1511  ;++
                               1511  ;
                               1511  ; FUNCTIONAL DESCRIPTION:
                               1511  ;
                               1511  ;      Issues calls to various SCRPKG routines to display screen
                               1511  ;      output for the PROCESSES class (regular display).
                               1511  ;
                               1511  ; INPUTS:
                               1511  ;
                               1511  ;       4(AP) - address of CDB (Class Descriptor Block) pointer
                               1511  ;               for the PROCESSES class.
                               1511  ;
                               1511  ;       8(AP) - address of quadword containing the system time
                               1511  ;               value of the latest collection buffer.
                               1511  ;
                               1511  ; IMPLICIT INPUTS:
                               1511  ;
                               1511  ;      MCAPTR - pointer to MCA (Monitor Communication Area)
                               1511  ;      MRBPTR - pointer to MRB (Monitor Request Block)
                               1511  ;      SPTR   - pointer to SYI (System Information Area)
                               1511  ;
                               1511  ;      MCA$L_PROC_DISP - longword containing number of processes
                               1511  ;                        to display this interval.
                               1511  ;
                               1511  ;      PREV_PD         - longword containing number of processes
                               1511  ;                        displayed for the previous interval.
                               1511  ;
                               1511  ;
                               1511  ; OUTPUTS:
                               1511  ;
                               1511  ;      None
                               1511  ;
                               1511  ; IMPLICIT OUTPUTS:
                               1511  ;
                               1511  ;      Entire display buffer (FAOSTK) displayed to SYS$OUTPUT for
                               1511  ;      this display event.
                               1511  ;
                               1511  ;      PREV_PD -- longword set to the number of processes displayed
                               1511  ;                 this interval, for use next time through.
                               1511  ;
                               1511  ; ROUTINE VALUE:
                               1511  ;
                               1511  ;      R0 = SS$_NORMAL, or screen package error status.
                               1511  ;
                               1511  ; SIDE EFFECTS:
                               1511  ;
                               1511  ;      none
                               1511  ;
                               1511  ;--
                               1511
                         0FFC  1511  .ENTRY  DISPLAY_PROCS, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                               1513
          56    04 BC    D0    1513         MOVL    @4(AP),R6               ; Load CDB pointer
    5B    00000000'EF    D0    1517         MOVL    MCAPTR,R11              ; Load MCA pointer
          5A    2E A6    D0    151E         MOVL    CDB$A_BUFFERS(R6),R10   ; Load address of buffer block
```

```
   55    08 AA   D0  1522  4929          MOVL    MBP$A_PR_FAOSTK(R10),R5 ; Get pointer to FAO stack
   57    18 AB   D0  1526  4930          MOVL    MCA$L_PROC_DISP(R11),R7 ; Get number of procs to display
            03   12  152A  4931          BNEQ    10$                     ; Continue if we have some
          0147   31  152C  4932          BRW     110$                    ; Get out quickly if none to display
```

```
                              152F  ;
                              152F  ; Compute number of lines to erase (difference in the number of
                              152F  ; processes from previous display to this one).
                              152F  ;
                              152F  ; Clear the display area if necessary.
                              152F  ;
                              152F
                              152F  10$:
                 59    D4     152F             CLRL    R9                          ; Assume no lines to erase
      28 32 AB   06    E1     1531             BBC     #MCA$V_ERA_SCRL,MCA$W_FLAGS(R11),30$
                              1536                                                 ; If bit clear, no need to erase
      0F    00E4'CF    D1     1536             CMPL    W^PREV_PD,#VTDATALINES      ; Previous display a single screen?
                 16    15     153B             BLEQ    20$                         ; Yes -- skip the erase
                              153D             ALLOC   6,R1,R3                     ; Get 6 bytes for call to BLINK
               01B2    30     154A             BSBW    BLINK                       ; Erase entire display area
              0E 50    E8     154D             BLBS    R0,30$                      ; Continue if status OK
               0135    31     1550             BRW     DPROCS_RET                  ; Return with status if failed
                              1553                                                 ; (already logged)
                              1553  20$:
            0F   57    D1     1553             CMPL    R7,#VTDATALINES             ; Current display a single screen?
                 06    14     1556             BGTR    30$                         ; No -- continue
   59   00E4'CF  57    C3     1558             SUBL3   R7,W^PREV_PD,R9             ; Yes -- calc lines to erase
                              155E
                              155E  ;
                              155E  ; Compute and format ASCII uptime for setup display line.
                              155E  ;
                              155E
                              155E  30$:
                              155E             ALLOC   8,R1,R2                     ; Allocate 8 stack bytes for time calcs
    54   00000000'EF   D0     156B             MOVL    SPTR,R4                     ; Get SYI pointer
         62   03 A4    7D     1572             MOVQ    MNR_SYI$Q_BOOTTIME(R4),(R2) ; Boot time into calc area
         50   08 BC    7D     1576             MOVQ    @8(AP),R0                   ; Get collection time
            62   50    C2     157A             SUBL2   R0,(R2)                     ; Subtract coll time from boot time
         04 A2   51    D9     157D             SBWC    R1,4(R2)                    ; ....
                              1581             ALLOC   13,R4,R1                    ; Get ASCTIM output buffer
                              158E             $ASCTIM_S TIMBUF=(R4), TIMADR=(R2), CVTFLG=#0 ; Get ascii uptime
              03 50    E8     159D             BLBS    R0,40$                      ; Continue if status OK
               00E6    31     15A0             BRW     DPROCS_ERR                  ; Log error & ret with status if failed
                              15A3
                              15A3  ;
                              15A3  ; Put out process count and uptime.
                              15A3  ;
                              15A3
                              15A3  40$:
            62   57    D0     15A3             MOVL    R7,(R2)                     ; Re-use R2 as ptr to $FAOL parm list
         04 A2   54    D0     15A6             MOVL    R4,4(R2)                    ; Insert uptime descr ptr into parm list
                              15AA             ALLOC   2,R1,R3                     ; Alloc 2 bytes for DISPLAY_PUT flags
            63   01    B0     15B7             MOVW    #1,(R3)                     ; Set bit to force DISPLAY_PUT thru $FAOL
                 52    DD     15BA             PUSHL   R2                          ; Push ptr to $FAOL parameter list
       0000261A'EF     DD     15BC             PUSHL   PROC_SETUP_STR+4            ; Push address of setup string
       00002616'EF     DF     15C2             PUSHAL  PROC_SETUP_STR              ; Push length of setup string
                 53    DD     15C8             PUSHL   R3                          ; Push DISPLAY_PUT request flags
  00001ADA'EF    04    FB     15CA             CALLS   #4,DISPLAY_PUT              ; Put out screen setup string
              03 50    E8     15D1             BLBS    R0,50$                      ; Continue if status OK
               00B1    31     15D4             BRW     DPROCS_RET                  ; Return with status if failed (already logg
                              15D7  4990 ;
                              15D7  4991 ; Loop putting out a line for each process
                              15D7  4992 ;
                              15D7  4993
                              15D7  4994 50$:
                              15D7  4995          ALLOC   8,R1,R2                  ; Allocate a descriptor
        62   0000004B 8F  D0  15E4  4996          MOVL    #PROC_LINE,(R2)          ; Move const len of FAO output buff
     04 A2   00001A07'EF  D0  15EB  4997          MOVL    OUTDSC+4,4(R2)           ; Load ptr to FAO output buffer
               58   57    D0  15F3  4998          MOVL    R7,R8                    ; Load no. of procs for this interval
                              15F6  4999 60$:
              0F    58    D1  15F6  5000          CMPL    R8,#VTDATALINES          ; Will processes fit in display area?
                    42    15  15F9  5001          BLEQ    80$                      ; Yes -- go put them there
                              15FB  5002
        0E 00000000'EF    E9  15FB  5003          BLBC    CTRLCZ_HIT,70$           ; No -- go fill screen if collecting
                              1602  5004                                           ; Collection has ended
        51   00000000'EF  D0  1602  5005          MOVL    MRBPTR,R1                ; Load MRB pointer
        02 43 A1    05    E0  1609  5006          BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),70$
                              160E  5007                                           ; Go fill screen if output to file
                    75    11  160E  5008          BRB     130$                     ; Quit displaying
                              1610  5009 70$:
              54    0F    D0  1610  5010          MOVL    #VTDATALINES,R4          ; Do a screenful
                  008F    30  1613  5011          BSBW    FILL_SCREEN              ; .....
                 70 50    E9  1616  5012          BLBC    R0,DPROCS_ERR            ; Exit if error
                              1619  5013
                  0071    30  1619  5014          BSBW    PRINT_SCREEN             ; Force SCRPKG to display screen
                 6A 50    E9  161C  5015          BLBC    R0,DPROCS_ERR            ; Exit if error
                              161F  5016
                  012A    30  161F  5017          BSBW    HOLD_SCREEN              ; Wait between screenfuls
                 64 50    E9  1622  5018          BLBC    R0,DPROCS_ERR            ; Exit if error
                              1625  5019
                              1625  5020          ALLOC   6,R1,R3                  ; Get 6 bytes for call to BLINK
                  00CA    30  1632  5021          BSBW    BLINK                    ; Erase entire display area
                 50 50    E9  1635  5022          BLBC    R0,DPROCS_RET            ; Return with status if failed
                              1638  5023                                           ; (already logged)
              58    0F    C2  1638  5024          SUBL2   #VTDATALINES,R8          ; Calculate remaining processes
                    B9    11  163B  5025          BRB     60$                      ; ... and go get them displayed
```

```
                          163D  5027 80$:
     0E 00000000      E9  163D  5028         BLBC    CTRLCZ_HIT,90$          ; Go fill screen unless CTRL-C or Z hit
                          1644  5029                                         ; Col.ection has ended
  51   00000000'EF    D0  1644  5030         MOVL    MRBPTR,R1               ; Load MRB pointer
     02 43 A1   05    E0  164B  5031         BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),90$
                          1650  5032                                         ; Go fill screen if output to file
                33    11  1650  5033         BRB     130$                    ; Quit displaying
                          1652  5034 90$:
          54    58    D0  1652  5035         MOVL    R8,R4                   ; Load no. of procs for final screenful
                4E    10  1655  5036         BSBB    FILL_SCREEN             ; ... and put them out
             2F 50    E9  1657  5037         BLBC    R0,DPROCS_ERR           ; Exit if error
                          165A  5038
                          165A  5039 ;
                          165A  5040 ; At this point, all process lines have been sent to the SCRPKG.
                          165A  5041 ;
                          165A  5042 ; If necessary, erase individual screen lines left over from
                          165A  5043 ; previous display event.
                          165A  5044 ;
                          165A  5045
          53    59    D0  165A  5046         MOVL    R9,R3                   ; Retrieve number of lines to erase
                17    15  165D  5047         BLEQ    110$                    ; Continue if none to erase
     54    58   08    C1  165F  5048         ADDL3   #FIRST_DATA_LINE,R8,R4  ; Compute 1st line to be erased
                          1663  5049 100$:
                01    DD  1663  5050         PUSHL   #1                      ; Always erase from column 1
                54    DD  1665  5051         PUSHL   R4                      ; ... at current row
  00000000'GF   02    FB  1667  5052         CALLS   #2,G^SCR$ERASE_LINE     ; Erase current line
             18 50    E9  166E  5053         BLBC    R0,DPROCS_ERR           ; Quit if error
                54    D6  1671  5054         INCL    R4                      ; Get next line number
             ED 53    F5  1673  5055         SOBGTR  R3,100$                 ; Go erase next line
```

```
                               1676  ;
                               1676  ; Save number of processes (this interval) for use in next interval.
                               1676  ;
                               1676
                               1676  110$:
         00E4'CF   57    D0    1676          MOVL    R7,W^PREV_PD            ; Remember number of procs for next interval
        00 32 AB   06    E2    167B          BBSS    #MCA$V_ERA_SCRL,MCA$W_FLAGS(R11),120$
                               1680                                          ; Indicate erase display area next int
                               1680  120$:
                   0B    10    1680          BSBB    PRINT_SCREEN            ; Force SCRPKG to display the screen
                04 50    E9    1682          BLBC    R0,DPROCS_ERR           ; Exit if error
                               1685  130$:
              50   01    D0    1685          MOVL    #SS$_NORMAL,R0          ; Successful status
                               1688
                               1688  DPROCS_RET:
                         04    1688          RET                             ; Return with status set
                               1689
                               1689  DPROCS_ERR:
                03A0     30    1689          BSBW    DISPERR                 ; Log display error
                         04    168C          RET                             ; Return with status
                               168D
                               168D
                               168D  ;
                               168D  ; PRINT_SCREEN subroutine.
                               168D  ;
                               168D  ; Forces SCRPKG to actually output entire screen.
                               168D  ;
                               168D
                               168D  PRINT_SCREEN:
                               168D
     00000000'GF   00    FB    168D          CALLS   #0,G^LIB$PUT_BUFFER     ; Output SCRPKG buffer and stop buffering
                0D 50    E9    1694          BLBC    R0,10$                  ; Exit if error
        000020AF'EF      7F    1697          PUSHAQ  SCRDSC                  ; Push MONITOR buffer addr
     00000000'GF   01    FB    169D          CALLS   #1,G^LIB$SET_BUFFER     ; Set buffering mode again
                               16A4  10$:
                         05    16A4          RSB                             ; Return with R0 set
                               16A5
```

```
16A5   ;
16A5   ; The FILL_SCREEN subroutine fills the display area with
16A5   ; the number of processes indicated in R4.
16A5   ;
16A5   ; REGISTER INPUTS:
16A5   ;
16A5   ;       R2 -- Pointer to an 8-byte stack area consisting of an FAO
16A5   ;               output descriptor.
16A5   ;               This register not modified by FILL_SCREEN.
16A5   ;
16A5   ;       R4 -- Number of processes to display (less than or equal
16A5   ;               to display area size).
16A5   ;               This register is destroyed by FILL_SCREEN.
16A5   ;
16A5   ;       R5 -- Pointer to current process in display buffer (FAO Stack).
16A5   ;               On exit, R5 is updated to point to the first process
16A5   ;               to be displayed on the next screen.
16A5   ;
16A5   ; SCRATCH REGISTERS:
16A5   ;
16A5   ;       R1
16A5   ;
16A5   ;       R3 --  Pointer to FAO control string
16A5   ;
16A5   ;       R10 -- Number of display lines to advance after PUT_LINE.
16A5   ;
16A5   ; RETURN REGISTER:
16A5   ;
16A5   ;       R0 -- On exit, contains status of most recent SCRPKG call.
16A5   ;
```

```
                        16A5  FILL_SCREEN:
                        16A5
              01    DD  16A5          PUSHL   #1                          ; Set cursor to
              08    DD  16A7          PUSHL   #FIRST_DATA_LINE            ; ... first PROCESSES
00000000'GF   02    FB  16A9          CALLS   #2,G^SCR$SET_CURSOR         ; ... display line
           4B 50    E9  16B0          BLBC    R0,40$                      ; Exit if error
                        16B3
         5A   01    D0  16B3          MOVL    #1,R10                      ; Load number of lines to advance
                        16B6  10$:
         54   01    D1  16B6          CMPL    #1,R4                       ; Only one process left to display?
              02    12  16B9          BNEQ    20$                         ; No -- continue
              5A    D4  16BB          CLRL    R10                         ; Yes -- indicate no advance
                        16BD
                        16BD  ;
                        16BD  ; Choose an FAO control string based on process residency
                        16BD  ;
                        16BD
                        16BD  20$:
53   00002631'EF    DE  16BD          MOVAL   PROC_RES_STR,R3             ; Assume this process is resident
           3C A5    D5  16C4          TSTL    MNR_PRO$K_FSIZE-4(R5)       ; Is last longword for this process zero?
              07    12  16C7          BNEQ    30$                         ; No -- process is resident
53   00002676'EF    DE  16C9          MOVAL   PROC_NRES_STR,R3            ; Yes -- process is non-resident
                        16D0  ;
                        16D0  ; Issue FAOL
                        16D0  ;
                        16D0
                        16D0  30$:
                        16D0          $FAOL_S CTRSTR=(R3), OUTBUF=OUTDSC, PRMLST=(R5)
           18 50    E9  16E3          BLBC    R0,40$                      ; Exit if error
                        16E6
                        16E6  ;
                        16E6  ; Send process output line to SCRPKG
                        16E6  ;
                        16E6
              5A    DD  16E6          PUSHL   R10                         ; Push number of lines to advance
              52    DD  16E8          PUSHL   R2                          ; Push addr of FAO output descriptor
00000000'GF   02    FB  16EA          CALLS   #2,G^SCR$PUT_LINE           ; Give one process line to SCRPKG
           0A 50    E9  16F1          BLBC    R0,40$                      ; Exit if error
                        16F4
55   00000040 8F    C0  16F4          ADDL2   #MNR_PRO$K_FSIZE,R5         ; Point to next process in FAO stack
           B8 54    F5  16FB          SOBGTR  R4,10$                      ; Loop back to do next process
                        16FE  40$:
                    05  16FE          RSB                                 ; Return
```

```
                             16FF  5170 ;
                             16FF  5171 ; BLINK Subroutine.
                             16FF  5172 ;
                             16FF  5173 ; Erases entire data display area.
                             16FF  5174 ; Also calls DISPLAY_PUT to replace the status
                             16FF  5175 ;       (footing) line if necessary.
                             16FF  5176 ;
                             16FF  5177 ; R3 = Address of 6-byte stack area for call to DISPLAY_PUT:
                             16FF  5178 ;
                             16FF  5179 ;       1) longword to hold length of status string;
                             16FF  5180 ;       2) pair of bytes used for request flags.
                             16FF  5181 ;
                             16FF  5182 ; Upon exit, R0 contains status from DISPLAY_PUT call.
                             16FF  5183 ;
                             16FF  5184 ; Register R1 is destroyed by this subroutine.
                             16FF  5185 ;
                             16FF  5186
                             16FF  5187 BLINK:
                  01     DD  16FF  5188         PUSHL   #1                      ; Column 1
                  08     DD  1701  5189         PUSHL   #FIRST_DATA_LINE        ; First line of data display area
   51   00000000'EF      D0  1703  5190         MOVL    MRBPTR,R1               ; Get MRB pointer
     03 43 A1    0B      E1  170A  5191         BBC     #MRB$V_MFSUM,MRB$W_FLAGS(R1),10$ ; Br if not m.f. summary
            6E   02      C0  170F  5192         ADDL2   #2,(SP)                 ; 1st data line lower for m.f. summary
                             1712  5193 10$:
   00000000'GF   02      FB  1712  5194         CALLS   #2,G^SCR$ERASE_PAGE     ; Erase to end of screen
                             1719  5195
                             1719  5196 ;
                             1719  5197 ; Now replace the footing line that was erased if necessary
                             1719  5198 ;
                             1719  5199
              50   01    D0  1719  5200         MOVL    #SS$_NORMAL,R0          ; Assume normal status
   51   00000000'EF      D0  171C  5201         MOVL    MRBPTR,R1               ; Get MRB pointer
     23 43 A1    0B      E0  1723  5202         BBS     #MRB$V_MFSUM,MRB$W_FLAGS(R1),20$
                             1728  5203                                         ; Don't refresh status if m.f. summary
        43 A1    0E      B3  1728  5204         BITW    #<MRB$M_PLAYBACK+MRB$M_SUMMARY+MRB$M_RECORD>,MRB$W_FLAGS(R1)
                             172C  5205                                         ; Any status fields need refreshing?
                  1D     13  172C  5206         BEQL    20$                     ; No -- go exit
             23F7'CF     DF  172E  5207         PUSHAL  W^STATUS_PARMS          ; Push addr of $FAOL parameter list
             2300'CF     9F  1732  5208         PUSHAB  W^STATUS_STR+1          ; Push address of status string
      63     22FF'CF     9A  1736  5209         MOVZBL  W^STATUS_STR,(R3)       ; Load status string length
                  53     DD  173B  5210         PUSHL   R3                      ; Push its address
        04 A3    01      B0  173D  5211         MOVW    #1,4(R3)                ; Set bit to force DISPLAY_PUT thru $FAOL
               04 A3     DF  1741  5212         PUSHAL  4(R3)                   ; Push ptr to DISPLAY_PUT request flags
   00001ADA'EF   04      FB  1744  5213         CALLS   #4,DISPLAY_PUT          ; Put out status string on bottom line
                             174B  5214 20$:
                         05  174B  5215         RSB                             ; Return with status in R0
```

```
                               174C  ;
                               174C  ; HOLD_SCREEN Subroutine.
                               174C  ;
                               174C  ;       Waits after a full screen has been displayed
                               174C  ;       in order to let the user see it before the
                               174C  ;       next screenful arrives.
                               174C  ;
                               174C  ; Upon exit, R0 contains status from $SETIMR or $WAITFR.
                               174C  ;
                               174C  ; Register R1 is destroyed by this subroutine.
                               174C  ;
                               174C
                               174C HOLD_SCREEN:
                               174C
          50   01         D0   174C         MOVL    #SS$_NORMAL,R0             ; Assume normal status
    31 00000000'EF        E8   174F         BLBS    CTRLCZ_HIT,10$             ; If CTRL-C or Z hit, don't hold
 51    00000000'EF        D0   1756         MOVL    MRBPTR,R1                  ; Get MRB pointer
     25 43 A1   05        E0   175D         BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),10$
                               1762                                            ; Don't hold if output to file
                               1762         $SETIMR_S EFN=#BE'_EV_FLAG, DAYTIM=VIEWING_DEL
                               1777                                            ; Set time between screens
             0D 50        E9   1777         BLBC    R0,10$                     ; Exit if error
                               177A
                               177A         $WAITFR_S EFN=#BET_EV_FLAG         ; Wait between screenfuls
                               1787 10$:
                          05   1787         RSB                                ; Return with status
```

```
                       1788            .SBTTL  DISPLAY_TOP - Put PROCESSES/TOP Display Output to Screen
                       1788
                       1788      ;++
                       1788      ;
                       1788      ; FUNCTIONAL DESCRIPTION:
                       1788      ;
                       1788      ;       Constructs FAO control string for a PROCESSES/TOP screen
                       1788      ;       and calls DISPLAY_PUT to send the screen to the SCRPKG.
                       1788      ;
                       1788      ; INPUTS:
                       1788      ;
                       1788      ;        4(AP) - address of CDB (Class Descriptor Block) pointer
                       1788      ;                for the PROCESSES class.
                       1788      ;
                       1788      ; IMPLICIT INPUTS:
                       1788      ;
                       1788      ;       FAOSTK - FAO parameter list for a TOP screen
                       1788      ;
                       1788      ;       TOP_PROCS - byte containing count (up to 8) of
                       1788      ;                   TOP processes to display.
                       1788      ;
                       1788      ; OUTPUTS:
                       1788      ;
                       1788      ;       None
                       1788      ;
                       1788      ; IMPLICIT OUTPUTS:
                       1788      ;
                       1788      ;       Entire display buffer (FAOSTK) displayed to SYS$OUTPUT for
                       1788      ;       this display event.
                       1788      ;
                       1788      ; ROUTINE VALUE:
                       1788      ;
                       1788      ;       R0 = SS$_NORMAL, or screen package error status.
                       1788      ;
                       1788      ; SIDE EFFECTS:
                       1788      ;
                       1788      ;       none
                       1788      ;
                       1788      ;--
                       1788
                 0E7C  1788      .ENTRY  DISPLAY_TOP, ^M<R2,R3,R4,R5,R6,R9,R10,R11>
                       178A
 56    04 BC    D0     178A            MOVL    @4(AP),R6                 ; Load CDB pointer
 5B    04 A6    D0     178E            MOVL    CDB$A_FAOCTR(R6),R11      ; Load addr of FAO control string
                       1792
                       1792      ;
                       1792      ; Loop which concatenates as many FAO control string segments
                       1792      ; as needed to build the portion of the control string for
                       1792      ; processes to be displayed. The portion for null lines is
                       1792      ; built later.
                       1792      ;
                       1792
26C6'CF     08  90     1792            MOVB    #FIRST_DATA_LINE,W^TOPLNNO ; Load first TOP display line no
59   26C3'CF    9A     1797            MOVZBL  W^TOPSTR,R9               ; Load length of FAO ctr str for 1 line
5A   0022'CF    9A     179C            MOVZBL  W^TOP_PROCS,R10           ; Load number of TOP procs to display
           11   13     17A1            BEQL    20$                       ; Branch if none
                       17A3      10$:
```

```
      6B   26C4'CF    59    28  17A3   5301         MOVC3   R9,W^TOPSTR+1,(R11)      ; Move ctr str segment to ctr string
                 5B   59    C0  17A9   5302         ADDL2   R9,R11                   ; Point to next available byte in ctr str
           26C6'CF    02    80  17AC   5303         ADDB2   #2,W^TOPLNNO             ; Update cursor control to next disp line
                   EF 5A    F5  17B1   5304         SOBGTR  R10,10$                  ; Move one segment for each process
                                17B4   5305
                                17B4   5306 ;
                                17B4   5307 ; Now loop moving in control string segments for each
                                17B4   5308 ; null line to be displayed.
                                17B4   5309 ;
                                17B4   5310
                                17B4   5311 20$:
  5A    18 A6    0022'CF    83  17B4   5312         SUBB3   W^TOP_PROCS,CDB$L_ECOUNT(R6),R10 ; Calc number of null lines
                      1D    13  17B8   5313         BEQL    40$                      ; Branch if none
     26F2'CF     26C6'CF    90  17BD   5314         MOVB    W^TOPLNNO,W^ERLNNO       ; Line number of first null line
          59     26EF'CF    9A  17C4   5315         MOVZBL  W^ERLINE_STR,R9          ; Length of "erase line" control string
                                17C9   5316 30$:
      6B   26F0'CF    59    28  17C9   5317         MOVC3   R9,W^ERLINE_STR+1,(R11) ; Move ctr str segment to ctr string
                 5B   59    C0  17CF   5318         ADDL2   R9,R11                   ; Update control string pointer
           26F2'CF    02    80  17D2   5319         ADDB2   #2,W^ERLNNO              ; Update cursor control to next disp line
                   EF 5A    F5  17D7   5320         SOBGTR  R10,30$                  ; Move one segment for each null line
                                17DA   5321
                                17DA   5322 ;
                                17DA   5323 ; Call DISPLAY_PUT to put the screen image to SYS$OUTPUT.
                                17DA   5324 ;
                                17DA   5325
                                17DA   5326 40$:
                                17DA   5327         ALLOC   6,R1,R9                  ; Alloc 2 bytes for DISPLAY_PUT flags
                                17E7   5328                                          ; ... and a longword for ctr str size
                 69   01    90  17E7   5329         MOVB    #1,(R9)                  ; Set bit to force DISPLAY_PUT thru $FAOL
              01 A9   01    90  17EA   5330         MOVB    #1,1(R9)                 ; Force DISPLAY_PUT to output it now
    02 A9      5B   04 A6    C3  17EE   5331         SUBL3   CDB$A_FAOCTR(R6),R11,2(R9) ; Calc size of FAO control string
                                17F4   5332
              0103'CF       DF  17F4   5333         PUSHAL  W^FAOSTK                 ; Push ptr to $FAOL parameter list
                04 A6       DD  17F8   5334         PUSHL   CDB$A_FAOCTR(R6)         ; Push address of control string
                02 A9       DF  17FB   5335         PUSHAL  2(R9)                    ; Push length of control string
                   59       DD  17FE   5336         PUSHL   R9                       ; Push DISPLAY_PUT request flags
       00001ADA'EF    04    FB  1800   5337         CALLS   #4,DISPLAY_PUT           ; Put out screen setup string
                03 50       E9  1807   5338         BLBC    R0,DTOP_RET              ; Return with status if failed (already logg
                                180A   5339
                 50   01    D0  180A   5340         MOVL    #SS$_NORMAL,R0           ; Indicate success
                                180D   5341
                                180D   5342 DTOP_RET:
                        04      180D   5343         RET                              ; Return with status set
                        180E       .SBTTL  DISPLAY_HOMOG - Put Homog Class Display Output to Screen
                        180E
                        180E  ;++
                        180E  ;
                        180E  ; FUNCTIONAL DESCRIPTION:
                        180E  ;
                        180E  ;       Issues calls to various SCRPKG routines to display screen
                        180E  ;       output for the current (homogeneous) class.
                        180E  ;
                        180E  ; INPUTS:
                        180E  ;
                        180E  ;        4(AP) - address of CDB (Class Descriptor Block) pointer
                        180E  ;                for the current class.
                        180E  ;
                        180E  ; IMPLICIT INPUTS:
                        180E  ;
                        180E  ;       MCAPTR - pointer to MCA (Monitor Communication Area)
                        180E  ;       MRBPTR - pointer to MRB (Monitor Request Block)
                        180E  ;
                        180E  ;       CDX$L_PREV_DCT - longword containing number of elements
                        180E  ;                        displayed for the previous interval.
                        180E  ;
                        180E  ;
                        180E  ; OUTPUTS:
                        180E  ;
                        180E  ;       None
                        180E  ;
                        180E  ; IMPLICIT OUTPUTS:
                        180E  ;
                        180E  ;       Entire list of elements for this homogeneous class
                        180E  ;       displayed to SYS$OUTPUT for this display event.
                        180E  ;
                        180E  ;       CDX$L_PREV_DCT -- longword set to the number of elements displayed
                        180E  ;                 this interval, for use next time through.
                        180E  ;
                        180E  ; ROUTINE VALUE:
                        180E  ;
                        180E  ;       R0 = SS$_NORMAL, or screen package error status.
                        180E  ;
                        180E  ; SIDE EFFECTS:
                        180E  ;
                        180E  ;       none
                        180E  ;
                        180E  ;--
                        180E
                 09E8   180E  .ENTRY  DISPLAY_HOMOG, ^M<R3,R5,R6,R7,R8,R11>
                        1810
    56    04 BC   D0    1810          MOVL    @4(AP),R6                ; Load CDB pointer
    57    32 A6   D0    1814          MOVL    CDB$A_CDX(R6),R7         ; Load CDX pointer
5B  00000000'EF   D0    1818          MOVL    MCAPTR,R11               ; Load MCA pointer
                        181F          ALLOC   6,R1,R3                  ; Get 6 bytes for calls to BLINK
                        182C                                           ; ... and DISPLAY_PUT
                        182C
          28 A7   D5    182C          TSTL    CDX$A_DISPNAM(R7)        ; Check if we have a name display rtn
             03   12    182F          BNEQ    10$                      ; Br if we do
           009E   31    1831          BRW     80$                      ; Else, go exit if not
```

```
                                  1834  5402 10$:
            0F 32 AB    06    E1  1834  5403        BBC     #MCA$V_ERA_SCRL,MCA$W_FLAGS(R11),20$
                                  1839  5404                                                ; If bit clear, no need to erase
               0F    20 A7    D1  1839  5405        CMPL    CDX$L_PREV_DCT(R7), -          ; Previous display a single screen?
                                  183D  5406                #VTDATALINES
                          09  15  183D  5407        BLEQ    20$                            ; Yes -- skip the erase
                        FEBD  30  183F  5408        BSBW    BLINK                          ; Erase entire display area
                       03 50  E8  1842  5409        BLBS    R0,20$                         ; Continue if status OK
                        0097  31  1845  5410        BRW     DHOMOG_RET                     ; Return with status if failed
                                  1848  5411                                               ; (already logged)
                                  1848  5412 ;
                                  1848  5413 ; Display item name if necessary
                                  1848  5414 ;
                                  1848  5415
                                  1848  5416 20$:
            06 32 AB    06    E1  1848  5417        BBC     #MCA$V_ERA_SCRL,MCA$W_FLAGS(R11),30$
                                  184D  5418                                                ; If bit clear, need to display
               01    06 A7    91  184D  5419        CMPB    CDX$B_IDISCT(R7),#1            ; More than one item?
                          16  15  1851  5420        BLEQ    35$                            ; Br if no (no need to display)
                                  1853  5421 30$:
                                  1853  5422        ALLOC   8,R0,R1                        ; Alloc 2 lwords for FAOL parm stack
                                  1860  5423
                        0081  30  1860  5424        BSBW    DISP_HOM_ITMNAM                ; Display item name string
                                  1863  5425
                       03 50  E8  1863  5426        BLBS    R0,35$                         ; Continue if status OK
                        0076  31  1866  5427        BRW     DHOMOG_RET                     ; Return with status if failed
                                  1869  5428                                               ; (already logged)
```

```
                               1869  ;
                               1869  ; Loop putting out a screenful each time
                               1869  ;
                               1869
                               1869  35$:
        58   1C A7    D0       1869            MOVL    CDX$L_DCOUNT(R7),R8        ; Load no. of elements this interval
                55    D4       186D            CLRL    R5                         ; Init element index
                               186F  40$:
           0F   58    D1       186F            CMPL    R8,#VTDATALINES            ; Will elements fit on the screen?
                39    15       1872            BLEQ    60$                        ; Yes -- go put them there
                               1874
    0E 00000000'EF    E9       1874            BLBC    CTRLCZ_HIT,50$             ; No -- go fill screen if collecting
                               187B                                               ; Collection has ended
   51   00000000'EF   D0       187B            MOVL    MRBPTR,R1                  ; Load MRB pointer
     02 43 A1   05    E0       1882            BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),50$
                               1887                                               ; Go fill screen if output to file
                53    11       1887            BRB     90$                        ; Quit displaying
                               1889  50$:
                55    DD       1889            PUSHL   R5                         ; Push starting element index
                0F    DD       188B            PUSHL   #VTDATALINES               ; ... no. of elts to display
                56    DD       188D            PUSHL   R6                         ; ... and CDB address
                               188F
  00001934'EF   03    FB       188F            CALLS   #3,FILL_HOMOG_SCREEN       ; Display a screenful of elements
             46 50    E9       1896            BLBC    R0,DHOMOG_RET              ; Exit if error (already logged)
           55   0F    C0       1899            ADDL2   #VTDATALINES,R5            ; Compute element index for next fill
                               189C
              FEAD    30       189C            BSBW    HOLD_SCREEN                ; Wait between screenfuls
             3E 50    E9       189F            BLBC    R0,DHOMOG_ERR              ; Exit if error
                               18A2
              FE5A    30       18A2            BSBW    BLINK                      ; Erase entire display area
             37 50    E9       18A5            BLBC    R0,DHOMOG_RET              ; Return with status if failed
                               18A8                                               ; (already logged)
           58   0F    C2       18A8            SUBL2   #VTDATALINES,R8            ; Calculate remaining elements
                C2    11       18AB            BRB     40$                        ; ... and go get them displayed
```

```
                            18AD
                            18AD  ;
                            18AD  ; Fill a screen with remaining elements
                            18AD  ;
                            18AD
                            18AD  60$:
  0E 00000000'EF       E9   18AD          BLBC    CTRLCZ_HIT,70$            ; Go fill screen unless CTRL-C or Z hit
                            18B4                                            ; Collection has ended
51    00000000'EF      D0   18B4          MOVL    MRBPTR,R1                 ; Load MRB pointer
   02 43 A1            05   E0 18BB       BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),70$
                            18C0                                            ; Go fill screen if output to file
             1A        11   18C0          BRB     90$                       ; Quit displaying
                            18C2  70$:
             55        DD   18C2          PUSHL   R5                        ; Push starting element index
             58        DD   18C4          PUSHL   R8                        ; ... no. of elts to display
             56        DD   18C6          PUSHL   R6                        ; ... and CDB address
                            18C8
00001934'EF  03        FB   18C8          CALLS   #3,FILL_HOMOG_SCREEN      ; Display final screenful of elements
          0D 50        E9   18CF          BLBC    R0,DHOMOG_RET             ; Exit if error (already logged)
                            18D2
                            18D2  ;
                            18D2  ; At this point, all elements have been sent to the SCRPKG.
                            18D2  ;
                            18D2
```

```
                             18D2  ;
                             18D2  ; Save number of elements displayed this interval for use in next interval.
                             18D2  ;
                             18D2
                             18D2  80$:
20 A7    1C A7     D0       18D2          MOVL    CDX$L_DCOUNT(R7), -
                             18D7                  CDX$L_PREV_DCT(R7)       ; Remember no. of elts for next intv'l
00 32 AB    06     E2       18D7          BBSS    #MCA$V_ERA_SCRL,MCA$W_FLAGS(R11),90$
                             18DC                                           ; Indicate erase display area next int
                             18DC  90$:
      50    01     D0       18DC          MOVL    #SS$_NORMAL,R0           ; Successful status
                             18DF
                             18DF  DHOMOG_RET:
                   04       18DF          RET                              ; Return with status set
                             18E0
                             18E0  DHOMOG_ERR:
         0149      30       18E0          BSBW    DISPERR                  ; Log display error
                   04       18E3          RET                              ; Return with status
```

```
                            18E4  5509 ;
                            18E4  5510 ; DISP_HOM_ITMNAM Subroutine.
                            18E4  5511 ;
                            18E4  5512 ; Calls DISPLAY_PUT to display the item name string
                            18E4  5513 ; in the heading for this homogeneous class. It is
                            18E4  5514 ; entered into the SCRPKG buffer, but not actually
                            18E4  5515 ; output to the terminal.
                            18E4  5516 ;
                            18E4  5517 ; Upon input,
                            1dE4  5518 ;
                            18E4  5519 ; R1 = Address of 8-byte FAOL parm stack for call to DISPLAY_PUT
                            18E4  5520 ;
                            18E4  5521 ; R3 = Address of 6-byte stack area for call to DISPLAY_PUT:
                            18E4  5522 ;
                            18E4  5523 ;       1) longword to hold length of status string;
                            13E4  5524 ;       2) pair of bytes used for request flags.
                            18E4  5525 ;
                            18E4  5526 ; R6 = Address of CDB
                            18E4  5527 ; R7 = Address of CDX
                            18E4  5528 ;
                            18E4  5529 ; Upon exit, R0 contains status from DISPLAY_PUT call.
                            18E4  5530 ;
                            18E4  5531 ; Register R1 is destroyed by this subroutine.
                            18E4  5532 ;
                            18E4  5533
                            18E4  5534 DISP_HOM_ITMNAM:
                            18E4  5535
    2601'CF    06     90    18E4  5536          MOVB    #ILN_REG,W^ITMLNNO       ; Set up row no. for reg. displays
50   00000000'EF      D0    18E9  5537          MOVL    MRBPTR,R0                ; Get MRB pointer
   05 43 A0    0B     E1    18F0  5538          BBC     #MRB$V_MFSUM,MRB$W_FLAGS(R0),10$ ; Br if not m.f. summary
    2601'CF    02     82    18F5  5539          SUBB2   #2,W^ITMLNNO             ; Adjust row no. for m.f. summary
                            18FA  5540 10$:
     61   0000'CF     9A    18FA  5541          MOVZBL  W^NAME_COL,(R1)          ; Get col number for name string
               61     D7    18FF  5542          DECL    (R1)                     ; Express as additional spaces
      50   08 A7      9A    1901  5543          MOVZBL  CDX$B_IDISINDEX(R7),R0   ; Get item index for this disp event
                            1905  5544
     50   1C B640     9A    1905  5545          MOVZBL  @CDB$A_ITMSTR(R6)[R0],R0 ; Load IDB item number
            50   11     C4    190A  5546          MULL2   #IDB$K_ILENGTH,R0        ; Compute index into IDB table
   50   0000'CF40     9E    190D  5547          MOVAB   W^PERFTABLE[R0],R0       ; Address of IDB for this item
   04 A1   04 A0      D0    1913  5548          MOVL    IDB$A_LNAME(R0),4(R1)    ; Addr of item name str to FAOL stack
                            1918  5549
               51     DD    1918  5550          PUSHL   R1                       ; Push addr of FAOL parameter list
                            191A  5551
          25FF'CF     9F    191A  5552          PUSHAB  W^ITEM_NAM_STR+1         ; Push addr of item name FAOL ctrl str
     63   25FE'CF     9A    191E  5553          MOVZBL  W^ITEM_NAM_STR,(R3)      ; Load its length
               53     DD    1923  5554          PUSHL   R3                       ; Push address of length longword
      04 A3    01     B0    1925  5555          MOVW    #1,4(R3)                 ; Set bit to force DISPLAY_PUT thru $FAOL
            04 A3     DF    1929  5556          PUSHAL  4(R3)                    ; Push ptr to DISPLAY_PUT request flags
 00001ADA'EF   04     FB    192C  5557          CALLS   #4,DISPLAY_PUT           ; Put out item name in heading
                            1933  5558
                      05    1933  5559          RSB                              ; Return with status in R0
```

```
                        1934          .SBTTL  FILL_HOMOG_SCREEN - Fill a Screen with Homog Class Output
                        1934    ;++
                        1934    ;
                        1934    ; FUNCTIONAL DESCRIPTION:
                        1934    ;
                        1934    ;       Issues calls to DISPLAY_PUT to display a full screen
                        1934    ;       of output for this homogeneous class.
                        1934    ;
                        1934    ; INPUTS:
                        1934    ;
                        1934    ;        4(AP) - address of CDB (Class Descriptor Block)
                        1934    ;                for the current (homogeneous) class.
                        1934    ;
                        1934    ;        8(AP) - number of element names (e.g., disk names) to be displayed.
                        1934    ;
                        1934    ;       12(AP) - Element ID Table index of 1st element to be displayed.
                        1934    ;
                        1934    ;
                        1934    ; IMPLICIT INPUTS:
                        1934    ;
                        1934    ;       MRBPTR - pointer to MRB (Monitor Request Block)
                        1934    ;
                        1934    ;       MFSPTR - pointer to MFS (Multi-File Summary Block)
                        1934    ;
                        1934    ; OUTPUTS:
                        1934    ;
                        1934    ;       None
                        1934    ;
                        1934    ; IMPLICIT OUTPUTS:
                        1934    ;
                        1934    ;       Entire screen full of homogeneous class data, names and
                        1934    ;       heading information output to the terminal.
                        1934    ;
                        1934    ; ROUTINE VALUE:
                        1934    ;
                        1934    ;       R0 = SS$_NORMAL, or screen package error status.
                        1934    ;
                        1934    ; SIDE EFFECTS:
                        1934    ;
                        1934    ;       none
                        1934    ;
                        1934    ;--
                        1934
                  00F0  1934    .ENTRY  FILL_HOMOG_SCREEN, ^M<R4,R5,R6,R7>
                        1936
56    04 AC     D0      1936            MOVL    4(AP),R6                ; Load CDB pointer
57    32 A6     D0      193A            MOVL    CDB$A_CDX(R6),R7        ; Load CDX pointer
54    08 AC     D0      193E            MOVL    8(AP),R4                ; Load no. of elts to display
         03     12      1942            BNEQ    10$                     ; Br if have some
       00A7     31      1944            BRW     70$                     ; Else simply go output screen
                        1947    10$:
55    0C AC     D0      1947            MOVL    12(AP),R5               ; Load element index of first one
                        194B
                        194B    ;
                        194B    ; Set up call to names display routine.
                        194B    ;
                        194B
```

```
                    50  D4  194B  5618          CLRL    R0                          ; Init count of names to skip displaying
                    55  D5  194D  5619          TSTL    R5                          ; First screenful this interval?
                    0A  12  194F  5620          BNEQ    20$                         ; Br if not
        0F     20 A7    D1  1951  5621          CMPL    CDX$L_PREV_DCT(R7), -       ; Previous display a single screen?
                            1955  5622                  #VTDATALINES
                    04  14  1955  5623          BGTR    20$                         ; Br if not
        50     20 A7    D0  1957  5624          MOVL    CDX$L_PREV_DCT(R7),R0       ; Skip display of all "previous" names
                            195B  5625  20$:
              54    50  D1  195B  5626          CMPL    R0,R4                       ; Any additional names this interval?
                    2A  13  195E  5627          BEQL    40$                         ; Skip di'play if not
         7E    08   50  C1  1960  5628          ADDL3   R0,#FIRST_DATA_LINE,-(SP) ; Stack starting row number
     51   00000000'EF   D0  1964  5629          MOVL    MRBPTR,R1                   ; Get MRB pointer
        03 43 A1    0B  E1  196B  5630          BBC     #MRB$V_MFSUM,MRB$W_FLAGS(R1),30$ ; Br if not m.f. summary
              6E    02  C0  1970  5631          ADDL2   #2,(SP)                     ; 1st data line lower for m.f. summary
                            1973  5632  30$:
         7E    54   50  C3  1973  5633          SUBL3   R0,R4,-(SP)                 ; Stack name count
         7E    55   50  C1  1977  5634          ADDL3   R0,R5,-(SP)                 ; Stack element index of first ...
                            197B  5635                                              ; ... name to display
                    56  DD  197B  5636          PUSHL   R6                          ; Stack CDB address
                            197D  5637
                            197D  5638  ;
                            197D  5639  ; Call name display routine
                            197D  5640  ;
                            197D  5641
   00001A48'EF      04  FB  197D  5642          CALLS   #4,DISP_HOM_NAMES           ; Display the element names
                 03 50  E8  1984  5643          BLBS    R0,40$                      ; Br if OK status
                  009E  31  1987  5644          BRW     FHS_ERR                     ; Else go exit if error
                            198A  5645
                            198A  5646  ;
                            198A  5647  ; Now display the actual data
                            198A  5648  ;
                            198A  5649
                            198A  5650  ;
                            198A  5651  ; First, compute the length of the FAO control string
                            198A  5652  ;
                            198A  5653
                            198A  5654  40$:
        51     40 A6    9A  198A  5655          MOVZBL  CDB$B_FAOSEGLEN(R6),R1      ; Get length of the FAO segment
              51    54  C4  198E  5656          MULL2   R4,R1                       ; Compute length of FAO ctrl string
        50     41 A6    9A  1991  5657          MOVZBL  CDB$B_FAOPRELEN(R6),R0      ; Get length of the FAO prefix
         66    51   50  C1  1995  5658          ADDL3   R0,R1,CDB$L_FAOCTR(R6)      ; ... and add it in
                            1999  5659
                            1999  5660  ;
                            1999  5661  ; Alloc some space for DISPLAY_PUT flags
                            1999  5662  ;
                            1999  5663
                            1999  5664          ALLOC   2,R0,R4                     ; Alloc 2 bytes for DISPLAY_PUT flags
              64    01  90  19A6  5665          MOVB    #1,(R4)                     ; Set bit to force DISPLAY_PUT thru $FAOL
                 01 A4  94  19A9  5666          CLRB    1(R4)                       ; ... but don't force to screen yet
                            19AC  5667
                            19AC  5668  ;
                            19AC  5669  ; Calculate and stack beginning of FAOSTK segment for this screen
                            19AC  5670  ;
                            19AC  5671
     50   00000000'EF   D0  19AC  5672          MOVL    MRBPTR,R0                   ; Get MRB pointer
       0E 43 A0     0B  E1  19B3  5673          BBC     #MRB$V_MFSUM,MRB$W_FLAGS(R0),50$ ; Br if not m.f. summary
     50   00000000'EF   D0  19B8  5674          MOVL    MFSPTR,R0                   ; Get MFS pointer
```

```
      50    34 A0    04     C5   198F   5675              MULL3    #4,MFS$L_LWORDS(R0),R0    ; Compute bytes in FAOSTK for one elt
                     0C     11   19C4   5676              BRB      60$                       ; Go compute offset
                                 19C6   5677 50$:
               50    20     D0   19C6   5678              MOVL     #<4*TAB_LWORDS>,R0        ; Assume tabular display
         00    42 A6     91   19C9   5679              CMPB     CDB$B_ST(R6),#ALL_STAT    ; ALL statistic requested?
                     03     13   19CD   5680              BEQL     60$                       ; Br if so
                                 19CF   5681
               50    0C     D0   19CF   5682              MOVL     #<4*BAR_LWORDS>,R0        ; Bar graph display
                                 19D2   5683 60$:
               50    55     C4   19D2   5684              MULL2    R5,R0                     ; Compute offset to first data to display
  7E    00000103'EF40     9E   19D5   5685              MOVAB    L^FAOSTK[R0],-(SP)        ; ... and stack its address
                                 19DD   5686
               04 A6     DD   19DD   5687              PUSHL    CDB$A_FAOCTR(R6)          ; Stack address of FAO ctrl str
                     66     DF   19E0   5688              PUSHAL   CDB$L_FAOCTR(R6)          ; ... and its length
                     54     DD   19E2   5689              PUSHL    R4                        ; Stack DISPLAY_PUT request flags
                                 19E4   5690
     00001ADA'EF     04     FB   19E4   5691              CALLS    #4,DISPLAY_PUT            ; Put screenful of homog class data
                                 19EB   5692
               39 50     E9   19EB   5693              BLBC     R0,FHS_RET                ; Return with status if failed
                                 19EE   5694                                                 ; ... (already logged)
                                 19EE   5695
                                 19EE   5696 ;
                                 19EE   5697 ; Send "REGSET" escape sequence to screen
                                 19EE   5698 ; to get back to the regular character set,
                                 19EE   5699 ; and force all accumulated output to screen.
                                 19EE   5700 ;
                                 19EE   5701
                                 19EE   5702 70$:
                                 19EE   5703              ALLOC    2,R0,R4                   ; Alloc 2 bytes for DISPLAY_PUT flags
                     64     94   19FB   5704              CLRB     (R4)                      ; Indicate no $FAOL filter needed
         01 A4    01     90   19FD   5705              MOVB     #1,1(R4)                  ; Force all accumulated output to screen
                                 1A01   5706
                                 1A01   5707              ALLOC    2,R0,R1                   ; Get space for esc seq string & descr
      61    0000'8F     B0   1A0E   5708              MOVW     #REGSET,(R1)              ; Move in "reg set" escape sequence
               04 A0     DD   1A13   5709              PUSHL    4(R0)                     ; Stack address of string
                     60     DF   1A16   5710              PUSHAL   (R0)                      ; ... and its length
                     54     DD   1A18   5711              PUSHL    R4                        ; Stack DISPLAY_PUT request flags
                                 1A1A   5712
     00001ADA'EF     03     FB   1A1A   5713              CALLS    #3,DISPLAY_PUT            ; Set reg set and display whole screen
                                 1A21   5714
               03 50     E9   1A21   5715              BLBC     R0,FHS_RET                ; Return with status if failed
                                 1A24   5716                                                 ; ... (already logged)
                                 1A24   5717
               50    01     D0   1A24   5718              MOVL     #SS$_NORMAL,R0            ; Successful status
                                 1A27   5719
                                 1A27   5720 FHS_RET:
                            04   1A27   5721              RET                                ; Return with status set
                                 1A28   5722
                                 1A28   5723 FHS_ERR:
                  0001     30   1A28   5724              BSBW     DISPERR                   ; Log display error
                            04   1A2B   5725              RET                                ; Return with status
```

```
                              1A2C  ;
                              1A2C  ; DISPERR Subroutine.
                              1A2C  ;
                              1A2C  ; Entered when an error has occurred in a system service or
                              1A2C  ; other routine while attempting to display to the terminal.
                              1A2C  ; Upon entry, R0 contains the failing status code. This
                              1A2C  ; routine makes a call to MON_ERR which records a MONITOR
                              1A2C  ; error code of MNR$_DISPERR and a subordinate error code of
                              1A2C  ; that in R0. Then, upon exit, the MNR$_DISPERR status is
                              1A2C  ; placed in R0.
                              1A2C  ;
                              1A2C  ; Upon entry,
                              1A2C  ;
                              1A2C  ; R0 = Error status code from a system service or other routine
                              1A2C  ;
                              1A2C  ; Upon exit,
                              1A2C  ;
                              1A2C  ; R0 = MNR$_DISPERR status code
                              1A2C  ;
                              1A2C  ; Register R1 is destroyed by this subroutine.
                              1A2C  ;
                              1A2C
                              1A2C DISPERR:
                    50   DD   1A2C          PUSHL   R0                      ; Bad status on stack
                    6E   DF   1A2E          PUSHAL  (SP)                    ; Stack pointer to bad status
           00000000'8F   DD   1A30          PUSHL   #MNR$_DISPERR           ; Stack MONITOR failing status code
  00001EB6'EF       02   FB   1A36          CALLS   #2,MON_ERR              ; Log the error
             5E     04   C0   1A3D          ADDL2   #4,SP                   ; Pop original status
  50       00000000'8F   D0   1A40          MOVL    #MNR$_DISPERR,R0        ; Get new status to caller
                         05   1A47          RSB                             ; Return
```

```
                              1A48        .SBTTL  DISP_HOM_NAMES - Display Names for Homog Class
                              1A48
                              1A48   ;++
                              1A48   ;
                              1A48   ; FUNCTIONAL DESCRIPTION:
                              1A48   ;
                              1A48   ;        Issues calls to SCRPKG routines to display names
                              1A48   ;        of homogeneous elements for the current screen.
                              1A48   ;        The names are entered into the SCRPKG buffer, but
                              1A48   ;        are not actually output to the screen.
                              1A48   ;
                              1A48   ; INPUTS:
                              1A48   ;
                              1A48   ;         4(AP) - address of CDB (Class Descriptor Block)
                              1A48   ;                 for the current (homogeneous) class.
                              1A48   ;
                              1A48   ;         8(AP) - Element ID Table index of 1st element to be displayed.
                              1A48   ;
                              1A48   ;        12(AP) - number of element names (e.g., disk names) to be displayed.
                              1A48   ;
                              1A48   ;        16(AP) - screen row number on which to display first element.
                              1A48   ;
                              1A48   ; IMPLICIT INPUTS:
                              1A48   ;
                              1A48   ; OUTPUTS:
                              1A48   ;
                              1A48   ;        None
                              1A48   ;
                              1A48   ; IMPLICIT OUTPUTS:
                              1A48   ;
                              1A48   ;        All names for the current screen full of elements are sent
                              1A48   ;        to the SCkPKG.
                              1A48   ;
                              1A48   ; ROUTINE VALUE:
                              1A48   ;
                              1A48   ;        R0 = SS$_NORMAL, or screen package error status.
                              1A48   ;
                              1A48   ; SIDE EFFECTS:
                              1A48   ;
                              1A48   ;        none
                              1A48   ;
                              1A48   ;--
                              1A48
                        07FC  1A48   .ENTRY  DISP_HOM_NAMES, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
                              1A4A
56     04 AC      D0          1A4A           MOVL    4(AP),R6                ; Load CDB pointer
57     32 A6      D0          1A4E           MOVL    CDB$A_CDX(R6),R7        ; Load CDX pointer
59     0C AC      D0          1A52           MOVL    12(AP),R9               ; Get number of elements to display
                              1A56           ALLOC   40,R0,R3                ; Get 10 longwords for an FAO stack
                              1A63           ALLOC   10,R0,R2                ; Allocate a descriptor & a word
```

```
                            1A70     ;
                            1A70     ; Get Element ID Table address of first element to be displayed
                            1A70     ;
                            1A70
         58  09 A7    9A    1A70             MOVZBL  CDX$B_ELIDLEN(R7),R8      ; Get length of an element ID
     5A  58  08 AC    C5    1A74             MULL3   8(AP),R8,R10              ; Compute offset to 1st display elt
         51  0C A7    D0    1A79             MOVL    CDX$A_ELIDTABLE(R7),R1    ; Get addr of elt ID table
    50   00000000'EF  D0    1A7D             MOVL    MRBPTR,R0                 ; Get MRB pointer
      04 43 A0    0B  E1    1A84             BBC     #MRB$V_MFSUM,MRB$W_FLAGS(R0),10$ ; Br if not m.f. summary
         51  18 A7    D0    1A89             MOVL    CDX$A_SELIDTABLE(R7),R1 ; Get addr of super elt ID table
                            1A8D     10$:
             5A   51  C0    1A8D             ADDL2   R1,R10                    ; Compute addr of 1st display elt
                            1A90
                            1A90     ;
                            1A90     ; Get row and column numbers for first element name
                            1A90     ;
                            1A90
         54  10 AC    D0    1A90             MOVL    16(AP),R4                 ; Get first row number
      55   0000'CF    9A    1A94             MOVZBL  W^NAME_COL,R5             ; ... and column number
                            1A99
                            1A99     ;
                            1A99     ; Call class-specific routine to fill the FAO stack
                            1A99     ; for the current element.
                            1A99     ;
                            1A99     ; NOTE -- this routine expects:
                            1A99     ;
                            1A99     ;       R0,R1 = scratch
                            1A99     ;       R3  = address of 10-longword FAO stack
                            1A99     ;       R6  = address of CDB
                            1A99     ;       R7  = address of CDX
                            1A99     ;       R10 = address of current element ID
                            1A99     ;
                            1A99
                            1A99     20$:
              28 B7   16    1A99             JSB     @CDX$A_DISPNAM(R7)        ; Fill the FAO stack
                            1A9C
                            1A9C     ;
                            1A9C     ; FAO stack is set up. Issue the $FAOL and SCR$PUT_SCREEN calls
                            1A9C     ;
                            1A9C
                            1A9C             $FAOL_S CTRSTR=@CDX$A_DISPFAO(R7), OUTLEN=8(R2), -
                            1A9C                     OUTBUF=W^OUTDSC, PRMLST=(R3) ; Format an element name
                            1AAF
              27 50   E9    1AAF             BLBC    R0,DHN_RET                ; Exit if error
         62  08 A2    3C    1AB2             MOVZWL  8(R2),(R2)                ; Move actual text len to descr
    04 A2   1A07'CF   D0    1AB6             MOVL    W^OUTDSC+4,4(R2)          ; Move addr of text to descr
                            1ABC
                            1ABC     ;
                            1ABC     ; Push SCR$PUT_SCREEN arguments on stack and call it to display one name
                            1ABC     ;
                            1ABC
                 00   DD    1ABC             PUSHL   #0                        ; No special screen attributes
                 55   DD    1ABE             PUSHL   R5                        ; Column number
                 54   DD    1AC0             PUSHL   R4                        ; Row number
                 62   DF    1AC2             PUSHAL  (R2)                      ; Text descriptor
    00000000'GF  04   FB    1AC4             CALLS   #4,G^SCR$PUT_SCREEN       ; Put a name string to terminal
                            1ACB
    0B 50   E9  1ACB  5866          BLBC    R0,DHN_RET              ; Exit if error
                1ACE  5867
 5A    58   C0  1ACE  5868          ADDL2   R8,R10                  ; Point to next element for display
       54   D6  1AD1  5869          INCL    R4                      ; Point to next row number
    C3 59   F5  1AD3  5870          SOBGTR  R9,20$                  ; ... and go do it
                1AD6  5871
 50    01   D0  1AD6  5872          MOVL    #SS$_NORMAL,R0          ; Successful status
                1AD9  5873
                1AD9  5874 DHN_RET:
            04  1AD9  5875          RET                             ; Return with status set
                1ADA  5876
```

```
1ADA           .SBTTL  DISPLAY_PUT - Put Display Output to Screen
1ADA
1ADA   ;++
1ADA   ;
1ADA   ; FUNCTIONAL DESCRIPTION:
1ADA   ;
1ADA   ;       Puts DISPLAY output to SYS$OUTPUT (or any file) using
1ADA   ;       the Screen Package. Depending on the setting of an input
1ADA   ;       flag, DISPLAY_PUT will either send the supplied buffer
1ADA   ;       directly on to the screen package, or run it through $FAOL
1ADA   ;       before sending it. A second input flag indicates whether or
1ADA   ;       not to actually output the data sent to the screen package.
1ADA   ;
1ADA   ; INPUTS:
1ADA   ;
1ADA   ;        4(AP) - address of 2 contiguous bytes, each containing a flag:
1ADA   ;
1ADA   ;               Byte 0: If low bit set, use supplied buffer as input
1ADA   ;                       to $FAOL, and send the resultant buffer
1ADA   ;                       to the screen package. Otherwise, send
1ADA   ;                       the buffer directly on to the screen
1ADA   ;                       package.
1ADA   ;
1ADA   ;               Byte 1: If low bit set, issue screen package calls to
1ADA   ;                       actually output the data. Otherwise,
1ADA   ;                       no such calls are made and the screen
1ADA   ;                       package merely buffers all received data.
1ADA   ;
1ADA   ;        8(AP) - address of longword containing length of buffer to put.
1ADA   ;
1ADA   ;       12(AP) - address of buffer to put.
1ADA   ;
1ADA   ;       16(AP) - optional address of $FAOL parameter list.
1ADA   ;
1ADA   ; IMPLICIT INPUTS:
1ADA   ;
1ADA   ;       OUTDSC - quadword string descriptor for $FAOL output buffer.
1ADA   ;       SCRDSC - quadword string descriptor for buffer required by SCRPKG.
1ADA   ;
1ADA   ; OUTPUTS:
1ADA   ;
1ADA   ;       none
1ADA   ;
1ADA   ; IMPLICIT OUTPUTS:
1ADA   ;
1ADA   ;       Translated buffer sent to Screen Package.
1ADA   ;
1ADA   ; ROUTINE VALUE:
1ADA   ;
1ADA   ;       R0 = SS$_NORMAL, or screen package error status.
1ADA   ;
1ADA   ; SIDE EFFECTS:
1ADA   ;
1ADA   ;       none
1ADA   ;
1ADA   ;--
1ADA
```

```
                        0004  1ADA  5935 .ENTRY  DISPLAY_PUT,^M<R2>
                              1ADC  5936
   0F 04 BC    00        E0   1ADC  5937         BBS      #0,@4(AP),10$              ; Go do $FAOL call if requested
         0C AC           DD   1AE1  5938         PUSHL    12(AP)                     ; Otherwise, simply put buffer
         08 BC           DD   1AE4  5939         PUSHL    @8(AP)                     ; ....
00001B5D'EF    02        FB   1AE7  5940         CALLS    #2,PUT_TO_SCREEN           ; ....
               3F        11   1AEE  5941         BRB      20$                        ; Join common code
                              1AF0  5942 10$:
                              1AF0  5943         ALLOC    10,R1,R2                   ; Allocate a descriptor & a word
      62    08 BC        D0   1AFD  5944         MOVL     @8(AP),(R2)                ; Move in length of buffer
   04 A2    0C AC        D0   1B01  5945         MOVL     12(AP),4(R2)               ; ... and address of buffer
                              1B06  5946         $FAOL_S CTRSTR=(R2), OUTLEN=8(R2), OUTBUF=OUTDSC, PRMLST=@16(AP)
            3B 50        E9   1B1B  5947         BLBC     R0,DP_ERR                  ; Exit if error
     00001A07'EF         DD   1B1E  5948         PUSHL    OUTDSC+4                   ; Push output buffer address
      7E    08 A2        3C   1B24  5949         MOVZWL   8(R2),-(SP)                ; ... and its length
00001B5D'EF    02        FB   1B28  5950         CALLS    #2,PUT_TO_SCREEN           ; Put buffer to screen
                              1B2F  5951 20$:
            27 50        E9   1B2F  5952         BLBC     R0,DP_ERR                  ; Exit if error
      7E    04 BC        D0   1B32  5953         MOVL     @4(AP),-(SP)               ; Get parameter bytes on stack
   1A 01 AE    00        E1   1B36  5954         BBC      #0,1(SP),30$               ; Go exit if no output requested
00000000'GF    00        FB   1B3B  5955         CALLS    #0,G^LIB$PUT_BUFFER        ; Output SCRPKG buffer and stop buffering
            14 50        E9   1B42  5956         BLBC     R0,DP_ERR                  ; Exit if error
     000020AF'EF         7F   1B45  5957         PUSHAQ   SCRDSC                     ; Push MONITOR buffer addr
00000000'GF    01        FB   1B4B  5958         CALLS    #1,G^LIB$SET_BUFFER        ; Set buffering mode again
            04 50        E9   1B52  5959         BLBC     R0,DP_ERR                  ; Exit if error
                              1B55  5960 30$:
         50    01        D0   1B55  5961         MOVL     #SS$_NORMAL,R0             ; No failing status hit
                         04   1B58  5962         RET                                 ; Return with success
                              1B59  5963 DP_ERR:
             FED0        30   1B59  5964         BSBW     DISPERR                    ; Log display error
                         04   1B5C  5965         RET                                 ; Return with status
```

```
                                 1B5D          .SBTTL  PUT_TO_SCREEN - Translate escape seqs and issue PUT_SCREEN
                                 1B5D
                                 1B5D  ;++
                                 1B5D  ;
                                 1B5D  ; FUNCTIONAL DESCRIPTION:
                                 1B5D  ;
                                 1B5D  ;       Translate a buffer with imbedded escape sequences to Screen
                                 1B5D  ;       Package (SCRPKG) calls. The escape sequences present on
                                 1B5D  ;       input are defined above in the messages declaration section.
                                 1B5D  ;       They are generally VT52-style sequences, with a few minor
                                 1B5D  ;       changes. These are converted to general-case SCRPKG calls
                                 1B5D  ;       to accommodate any terminal. Within the buffer, strings of
                                 1B5D  ;       text between escape sequences are sent to SCRPKG with a
                                 1B5D  ;       SCR$PUT_SCREEN call.
                                 1B5D  ;
                                 1B5D  ; INPUTS:
                                 1B5D  ;
                                 1B5D  ;       4(AP) - length of buffer to translate (longword).
                                 1B5D  ;       8(AP) - address of buffer to translate.
                                 1B5D  ;
                                 1B5D  ; IMPLICIT INPUTS:
                                 1B5D  ;
                                 1B5D  ;       SYSOUT_TYPE - SYS$OUTPUT terminal type (byte).
                                 1B5D  ;
                                 1B5D  ; OUTPUTS:
                                 1B5D  ;
                                 1B5D  ;       none
                                 1B5D  ;
                                 1B5D  ; IMPLICIT OUTPUTS:
                                 1B5D  ;
                                 1B5D  ;       none
                                 1B5D  ;
                                 1B5D  ; ROUTINE VALUE:
                                 1B5D  ;
                                 1B5D  ;       R0 = Worst status received from SCRPKG.
                                 1B5D  ;
                                 1B5D  ; SIDE EFFECTS:
                                 1B5D  ;
                                 1B5D  ;       The entire buffer has been sent to the SCRPKG.
                                 1B5D  ;
                                 1B5D  ;--
                                 1B5D
                           000C  1B5D  .ENTRY  PUT_TO_SCREEN,^M<R2,R3>
                                 1B5F
              52   04 AC    7D   1B5F          MOVQ    4(AP),R2                ; get len & addr of buffer to translate
                                 1B63  SCANBUF:
       0000272F'EF     52   7D   1B63          MOVQ    R2,TXT_DESC             ; save descriptor of remaining buffer
            63     52  1B   3A   1B6A          LOCC    #ESC,R2,(R3)            ; scan for escape character
                   52  50   7D   1B6E          MOVQ    R0,R2                   ; use R2-R3 instead of R0-R1
       0000272F'EF     52   A2   1B71          SUBW    R2,TXT_LENGTH           ; compute length of text
                       15   13   1B78          BEQL    10$                     ; br if no text between esc sequences
  00000000'GF   0000271B'EF FA   1B7A          CALLG   PUTSCRARG,G^SCR$PUT_SCREEN ; put text string into SCRPKG buffer
                    07 50   E8   1B85          BLBS    R0,10$                  ; continue if status OK
       0000008D'EF     50   D0   1B88          MOVL    R0,PTS_STAT             ; else, remember it for later
```

```
                                  1B8F   6022 10$:
               52   02     A2     1B8F   6023          SUBW    #2,R2                         ; update length of remaining buffer
                    03     18     1B92   6024          BGEQ    20$                           ; keep going if more chars in buffer
                  00DC     31     1B94   6025          BRW     PTS_RET                       ; all done if slid off end
                                  1B97   6026 20$:
               53   02     C0     1B97   6027          ADDL    #2,R3                         ; update ptr to remaining buffer
      2737'CF   0A   FF A3 3A     1B9A   6028          LOCC    -1(R3),#ES_TAB_LEN,W^ESC_SEQ_TABLE ; get offset into tab for CASE
                    06     12     1BA1   6029          BNEQ    30$                           ; go do CASE for known esc sequences
                  00D5     30     1BA3   6030          BSBW    PUT_ESC_SEQ                   ; unknown esc seq ... just put out as text
                  00B9     31     1BA6   6031          BRW     CHEKRET                       ; join common code
                                  1BA9   6032 30$:
                                  1BA9   6033          CASE    R0,<CHEKBUF,PTS_ESCY,PTS_ESCU,PTS_ESCR,PTS_ESCK, -
                                  1BA9   6034                  PTS_ESCJ,PTS_ESCH,PTS_ESCG,PTS_ESCF,PTS_ESCB,PTS_ESCL>,W
                    56     11     1BC3   6035          BRB     PTS_ESCH                      ; if out of range, do a cursor home
                                  1BC5   6036
                                  1BC5   6037 PTS_ESCB:                                      ; set "bold" attribute
      00 272B'CF    00     E2     1BC5   6038          BBSS    #SCR$V_BOLD,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib mask
                                  1BCB   6039 10$:
                  009E     31     1BCB   6040          BRW     CHEKBUF                       ; join common code
                                  1BCE   6041
                                  1BCE   6042 PTS_ESCL:                                      ; set "underline" attribute
         2615'CF    00     91     1BCE   6043          CMPB    #DEC_CRT,W^SYSOUT_TYPE        ; is SYS$OUTPUT device VT100-compat ?
                    06     12     1BD3   6044          BNEQU   10$                           ; no -- don't request underlining
      00 272B'CF    03     E2     1BD5   6045          BBSS    #SCR$V_UNDERLINE,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib
                                  1BDB   6046 10$:
                  008E     31     1BDB   6047          BRW     CHEKBUF                       ; join common code
                                  1BDE   6048
                                  1BDE   6049 PTS_ESCR:                                      ; set "reverse video" attribute
      00 272B'CF    01     E2     1BDE   6050          BBSS    #SCR$V_REVERSE,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib m
                                  1BE4   6051 10$:
                  0085     31     1BE4   6052          BRW     CHEKBUF                       ; join common code
                                  1BE7   6053
                                  1BE7   6054 PTS_ESCU:                                      ; clear all VT100 attribute settings
               272B'CF     D4     1BE7   6055          CLRL    W^ATTRIBMSK                   ; do it......
                  007E     31     1BEB   6056          BRW     CHEKBUF                       ; join common code
                                  1BEE   6057
                                  1BEE   6058 PTS_ESCY:                                      ; position cursor
               52   02     A2     1BEE   6059          SUBW    #2,R2                         ; update buffer length
                    03     18     1BF1   6060          BGEQ    10$                           ; continue if more buffer left
                  007D     31     1BF3   6061          BRW     PTS_RET                       ; err if no coordinates -- just quit
                                  1BF6   6062 10$:
            7E   01 A3     9A     1BF6   6063          MOVZBL  1(R3),-(SP)                   ; stack column number
               7E   63     9A     1BFA   6064          MOVZBL  (R3),-(SP)                    ; stack row number
               53   02     C0     1BFD   6065          ADDL    #2,R3                         ; update ptr to remaining buffer
       00000000'GF  02     FB     1C00   6066          CALLS   #2,G^SCR$SET_CURSOR           ; set the cursor position
                    59     11     1C07   6067          BRB     CHEKRET                       ; join common code
                                  1C09   6068
                                  1C09   6069 PTS_ESCK:                                      ; erase to end of line
       00000000'GF  00     FB     1C09   6070          CALLS   #0,G^LIB$ERASE_LINE           ; do exactly that
                    50     11     1C10   6071          BRB     CHEKRET                       ; join common code
                                  1C12   6072
                                  1C12   6073 PTS_ESCJ:                                      ; erase to end of page (screen)
       00000000'GF  00     FB     1C12   6074          CALLS   #0,G^LIB$ERASE_PAGE           ; do it
                    47     11     1C19   6075          BRB     CHEKRET                       ; join common code
                                  1C1B   6076
                                  1C1B   6077 PTS_ESCH:                                      ; cursor to home
                    01     DD     1C1B   6078          PUSHL   #1                            ; stack column number
```

```
                            01  DD  1C1D               PUSHL   #1                       ; stack row number
          00000000'GF       02  FB  1C1F               CALLS   #2,G^SCR$SET_CURSOR      ; position cursor to home
                            3A  11  1C26               BRB     CHEKRET                  ; join common code
                                    1C28
                                    1C28 PTS_ESCF:                                      ; select "alternate" graphics set
     0000270D'EF    00002702'EF 7E  1C28               MOVAQ   VT100_ALTSET,VT100_CURSET ; make the alternate set current
                            0B  11  1C33               BRB     SELECT_SET               ; ... and go output esc seq to select it
                                    1C35
                                    1C35 PTS_ESCG:                                      ; select "regular" graphics set
     0000270D'EF    000026F6'EF 7E  1C35               MOVAQ   VT100_REGSET,VT100_CURSET ; make the regular set current
                                    1C40
                                    1C40 SELECT_SET:
               2615'CF      00  91  1C40               CMPB    #DEC_CRT,W^SYSOUT_TYPE   ; is SYS$OUTPUT device VT100-compat ?
                            0F  12  1C45               BNEQU   10$                      ; no --try another type
               0000270D'EF      DD  1C47               PUSHL   VT100_CURSET             ; yes -- push addr of esc seq
          00000000'GF       01  FB  1C4D               CALLS   #1,G^SCR$PUT_SCREEN      ; ... and write it
                            0C  11  1C54               BRB     CHEKRET                  ; join common return from CASE
                                    1C56 10$:
          00002615'EF       01  91  1C56               CMPB    #VT5X,SYSOUT_TYPE        ; is it VT5x series ?
                            0D  12  1C5D               BNEQU   CHEKBUF                  ; no -- no need to change char set
                                    1C5F
                          0019  30  1C5F               BSBW    PUT_ESC_SEQ              ; write out the esc seq just scanned
                                    1C62
                                    1C62 CHEKRET:
                         07 50  E8  1C62               BLBS    R0,CHEKBUF               ; continue if status OK
          0000008D'EF       50  D0  1C65               MOVL    R0,PTS_STAT              ; else, remember it for later
                                    1C6C
                                    1C6C CHEKBUF:                                       ; common return point for CASE
                            52  B5  1C6C               TSTW    R2                       ; whole buffer examined yet ?
                            03  13  1C6E               BEQL    PTS_RET                  ; yes -- get out
                          FEF0  31  1C70               BRW     SCANBUF                  ; no -- go look at more
                                    1C73
                                    1C73 PTS_RET:
          50    0000008D'EF     D0  1C73               MOVL    PTS_STAT,R0              ; return status value
                                04  1C7A               RET
                                    1C7B
                                    1C7B
                                    1C7B PUT_ESC_SEQ:                                   ; subroutine to put an imbedded esc
                                    1C7B                                                ; ... sequence directly to the screen
                                    1C7B
          0000272F'EF       02  B0  1C7B               MOVW    #2,TXT_LENGTH            ; load length of esc sequence
       00002733'EF       FE A3  9E  1C82               MOVAB   -2(R3),TXT_START         ; load starting address
               0000272F'EF      7F  1C8A               PUSHAQ  TXT_DESC                 ; push descriptor addrress
          00000000'GF       01  FB  1C90               CALLS   #1,G^SCR$PUT_SCREEN      ; ... and put out "as is"
                                05  1C97               RSB
```

```
1C98           .SBTTL  SELECT_REV_LEVS - Select Revision Levels
1C98
1C98    ;++
1C98    ;
1C98    ; FUNCTIONAL DESCRIPTION:
1C98    ;
1C98    ;
1C98    ;       This routine is called from the REQUEST_INIT routine in
1C98    ;       REQUEST.PLI to select the appropriate Revision Level for
1C98    ;       each class being monitored. Once the level is selected,
1C98    ;       this routine stores the level number in REVLEVELS, a
1C98    ;       128-byte vector which contains level numbers for all
1C98    ;       classes being monitored. Then it moves the CHange
1C98    ;       Descriptor (CHD) for each class into its Class Descriptor
1C98    ;       Block (CDB/CDX). For playback, a class which has a revision
1C98    ;       level unknown to this version of MONITOR is flagged in
1C98    ;       the UNK_CLASSES vector.
1C98    ;
1C98    ; INPUTS:
1C98    ;
1C98    ;        4(AP) - address of a 128-bit vector describing classes
1C98    ;                to monitor. If class n is to be monitored,
1C98    ;                bit n is a 1; otherwise it is 0.
1C98    ;
1C98    ;        8(AP) - address of a 128-bit vector which will describe
1C98    ;                classes with revision levels unknown to this
1C98    ;                version of MONITOR (UNK_CLASSES). It is used only
1C98    ;                for playback requests. For live requests and playback
1C98    ;                of Version 3 files (all classes at Rev 0), 8(AP)
1C98    ;                contains 0. Upon entry, all bits are
1C98    ;                indeterminate. For each class n to be monitored,
1C98    ;                bit n is set to 0 if its revision level is known
1C98    ;                and 1 if its revision level is unknown.
1C98    ;
1C98    ;       12(AP) - address of HDR$T_REVLEVELS, a 128-byte vector
1C98    ;                indicating the revision level of each recorded
1C98    ;                (i.e., input) class (for playback requests only).
1C98    ;                For live requests and playback of Version 3 files
1C98    ;                (all classes at Rev 0), 12(AP) contains 0.
1C98    ;
1C98    ;       16(AP) - address of REVLEVELS, a 128-byte vector, into which
1C98    ;                will be stored a level number for each class being
1C98    ;                monitored. Upon input, all bytes contain 0.
1C98    ;
1C98    ; IMPLICIT INPUTS:
1C98    ;
1C98    ;       MAX_CLASS_NO -  maximum class number defined.
1C98    ;       MRBPTR -        pointer to MRB (Monitor Request Block)
1C98    ;       CDBHEAD -       table of contiguous CDBs.
1C98    ;
1C98    ; OUTPUTS:
1C98    ;
1C98    ;       For each class to be monitored, one of two things happens:
1C98    ;
1C98    ;               1) if its revision level is unknown, the appropriate
1C98    ;                       bit in UNK_CLASSES is set. (Can happen only
1C98    ;                       on playback); or,
                                1C98  6182 ;
                                1C98  6183 ;              2) the CHD (CHange Descriptor) is moved to the CDB/CDX.
                                1C98  6184 ;
                                1C98  6185 ;        Also, for each class to be monitored, the appropriate byte
                                1C98  6186 ;              in REVLEVELS is set to the selected revision level.
                                1C98  6187 ;
                                1C98  6188 ; IMPLICIT OUTPUTS:
                                1C98  6189 ;
                                1C98  6190 ;        None
                                1C98  6191 ;
                                1C98  6192 ; ROUTINE VALUE:
                                1C98  6193 ;
                                1C98  6194 ;        NORMAL
                                1C98  6195 ;
                                1C98  6196 ; SIDE EFFECTS:
                                1C98  6197 ;
                                1C98  6198 ;        None
                                1C98  6199 ;
                                1C98  6200 ;--
                                1C98  6201
                          03FC  1C98  6202 .ENTRY  SELECT_REV_LEVS, ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                                1C9A  6203
          57   00000000'EF   D0  1C9A  6204         MOVL    MRBPTR,R7                  ; Get MRB pointer for later use
                     08 AC   D5  1CA1  6205         TSTL    8(AP)                      ; Check if UNK_CLASSES is provided
                        08   13  1CA4  6206         BEQL    10$                        ; If not, don't reference it
 08 BC    10    00   FE AF 00   2C  1CA6  6207         MOVC5   #0,,#0,#16,@8(AP)          ; Assume all classes are NOT unknown
                                1CAE  6208
                                1CAE  6209 ;
                                1CAE  6210 ; Use FFS instruction to select classes to be monitored.
                                1CAE  6211 ;
                                1CAE  6212
                                1CAE  6213 10$:
                        55   D4  1CAE  6214         CLRL    R5                         ; Init starting bit position
                                1CB0  6215 20$:
                   53   20   D0  1CB0  6216         MOVL    #32,R3                     ; Init bit field size
                                1CB3  6217                                            ; NOTE -- must handle in 32-bit chunks
                   52   55   D0  1CB3  6218         MOVL    R5,R2                      ; Init start position of next chunk
                                1CB6  6219 30$:
    54    04 BC    53   52   EA  1CB6  6220         FFS     R2,R3,@4(AP),R4            ; Search class bits for next class no.
                                1CBC  6221                                            ; R4 contains class no. if found
                        0E   13  1CBC  6222         BEQL    40$                        ; Branch if none found this chunk
                        1C   10  1CBE  6223         BSBB    SELECT_REV                 ; Select Rev Level for this class
                   53   52   C0  1CC0  6224         ADDL2   R2,R3                      ; Compute next starting
              52   54   01   C1  1CC3  6225         ADDL3   #1,R4,R2                   ; ... position and field size
                   53   52   C2  1CC7  6226         SUBL2   R2,R3                      ; ... for this chunk
                        EA   11  1CCA  6227         BRB     30$                        ; Go search rest of chunk
                                1CCC  6228 40$:
    FFDC 55    20   0000'8F   3D  1CCC  6229         ACBW    #MAX_CLASS_NO,#32,R5,20$ ; Loop to process next chunk
                                1CD4  6230
          50   00000000'EF   D0  1CD4  6231         MOVL    NORMAL,R0                  ; Set normal status
                             04  1CDB  6232         RET                                ; Return
                                1CDC  6233
                                1CDC  6234
                                1CDC  6235 SELECT_REV:                                ; Select Rev Level for this class
                                1CDC  6236                                            ; NOTE -- R4 contains class number
                                1CDC  6237                                            ; Regs R2, R3, R4, R5 must not be changed
                                1CDC  6238
```

```
  56   54   00000053 8F    C5  1CDC  6239          MULL3   #CDB$K_SIZE,R4,R6          ; Compute offset to desired CDB
     56   00000000'EF46    9E  1CE4  6240          MOVAB   CDBHEAD[R6],R6             ; Index to CDB address
                               1CEC  6241
          0B 43 A7    03   E0  1CEC  6242          BBS     #MRB$V_PLAYBACK,MRB$W_FLAGS(R7),10$ ; If PLAYBACK, go do it
                               1CF1  6243                                             ; Else, stay here and do LIVE
             58   4F A6    D0  1CF1  6244          MOVL    CDB$A_CHDHDR(R6),R8        ; Get ptr to CHD header
                59    88   9A  1CF5  6245          MOVZBL  (R8)+,R9                   ; Get current rev level
                      4F   10  1CF8  6246          BSBB    MOVE_CHD                   ; ... and move CHD for it to CDB/CDX
                      4C   11  1CFA  6247          BRB     SR_RSB                     ; All done with this class
                               1CFC  6248
                               1CFC  6249 10$:                                        ; Playback
                               1CFC  6250
             58   4F A6    D0  1CFC  6251          MOVL    CDB$A_CHDHDR(R6),R8        ; Get ptr to CHD header
                50    88   9A  1D00  6252          MOVZBL  (R8)+,R0                   ; Get curr level from CHDHDR
                               1D03  6253 20$:
             59   0C AC    D0  1D03  6254          MOVL    12(AP),R9                  ; Get addr of recorded rev levels
                      3D   13  1D07  6255          BEQL    60$                        ; Br if none (recorded lev is 0)
              59   6944    9A  1D09  6256          MOVZBL  (R9)[R4],R9                ; Get recorded rev level
                50    59   D1  1D0D  6257          CMPL    R9,R0                      ; Recorded rev level greater than curr?
                      34   1B  1D10  6258          BLEQU   60$                        ; Br if no (and use recorded level)
          00 08 BC    54   E2  1D12  6259          BBSS    R4,@8(AP),30$              ; Yes -- set bit for this class ...
                               1D17  6260 30$:                                        ; ... in UNK_CLASSES
  28 04 BC   00000000'8F   E1  1D17  6261          BBC     #SYSTEM_CLSNO,@4(AP),SR_RSB ; Skip checks if no SYSTEM
        00000000'8F   54   D1  1D20  6262          CMPL    R4,#PROCS_CLSNO            ; Is this the PROCESSES class?
                      12   13  1D27  6263          BEQL    40$                        ; Br if yes
        0000000C'8F   54   D1  1D29  6264          CMPL    R4,#STATES_CLSNO           ; Is this the STATES class?
                      09   13  1D30  6265          BEQL    40$                        ; Br if yes
        00000000'8F   54   D1  1D32  6266          CMPL    R4,#MODES_CLSNO            ; Is this the MODES class?
                      0D   12  1D39  6267          BNEQ    SR_RSB                     ; Br if no -- all done with this class
                               1D3B  6268 40$:
  00 08 BC   00000000'8F   E2  1D3B  6269          BBSS    #SYSTEM_CLSNO,@8(AP),50$ ; Yes -- also set bit for SYSTEM class
                               1D44  6270 50$:                                        ; ... in UNK_CLASSES
                      02   11  1D44  6271          BRB     SR_RSB                     ; All done with this class
                               1D46  6272 60$:
                      01   10  1D46  6273          BSBB    MOVE_CHD                   ; Move CHD for this class to CDB
                               1D48  6274
                               1D48  6275 SR_RSB:
                           05  1D48  6276          RSB                                ; Return to caller
                               1D49  6277
                               1D49  6278
                               1D49  6279 MOVE_CHD:                                   ; Move CHD for selected rev level ...
                               1D49  6280                                             ; ... to the CDB
                               1D49  6281 ;
                               1D49  6282 ; Upon input,
                               1D49  6283 ;
                               1D49  6284 ;        R4 = the current class number,
                               1D49  6285 ;        R6 = addr of CDB for this class.
                               1D49  6286 ;        R8 = addr of first CHD,
                               1D49  6287 ;        R9 = the selected Rev Level,
                               1D49  6288 ;
                               1D49  6289 ; This routine alters R0 and R9.
                               1D49  6290 ;
                               1D49  6291
          10 BC44     59   90  1D49  6292          MOVB    R9,@16(AP)[R4]             ; Set revision level
              59      0D   C4  1D4E  6293          MULL2   #CHD$K_SIZE,R9             ; Compute offset to desired CHD
            59   6849      9E  1D51  6294          MOVAB   (R8)[R9],R9                ; R9 gets addr of desired CHD
            14 A6     69   D0  1D55  6295          MOVL    CHD$L_ICOUNT(R9),CDB$L_ICOUNT(R6) ; Move in item count
```

```
1C A6    04 A9   D0  1D59  6296            MOVL    CHD$A_ITMSTR(R9),CDB$A_ITMSTR(R6) ; ... and item string ptr
20 A6    08 A9   B0  1D5E  6297            MOVW    CHD$W_BLKLEN(R9),CDB$W_BLKLEN(R6) ; ... and block length
36 A6    0B A9   B0  1D63  6298            MOVW    CHD$W_DISPCTL(R9),CDB$W_DISPCTL(R6) ;.. and display ctl string
09 4B A6    05   E1  1D68  6299            BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$  ; Br if heterogeneous
      50  32 A6  D0  1D6D  6300            MOVL    CDB$A_CDX(R6),R0        ; Homogeneous class -- get CDX
09 A0    0A A9   90  1D71  6301            MOVB    CHD$B_ELIDLEN(R9),CDX$B_ELIDLEN(R0) ; Move in elem ID length
                     1D76  6302 10$:
                 05  1D76  6303            RSB                                     ; Return
                 1D77          .SBTTL  ESTAB_CTRLCZ - Establish CTRL-C,Z Handlers
                 1D77
                 1D77     ;++
                 1D77     ;
                 1D77     ; FUNCTIONAL DESCRIPTION:
                 1D77     ;
                 1D77     ;        This routine is called to set up a CTRL-C handler and a
                 1D77     ;        CTRL-Z handler for the SYS$COMMAND terminal device. A
                 1D77     ;        channel is assigned to SYS$COMMAND and its device class
                 1D77     ;        is checked for TERMINAL. If not terminal class, the
                 1D77     ;        handlers are not established.
                 1D77     ;
                 1D77     ;        Then $QIOW's are issued to the terminal driver to establish
                 1D77     ;        both handlers. If any system service call fails,
                 1D77     ;        the failing status is returned; otherwise, NORMAL status
                 1D77     ;        is returned.
                 1D77     ;
                 1D77     ; INPUTS:
                 1D77     ;
                 1D77     ;        None
                 1D77     ;
                 1D77     ; IMPLICIT INPUTS:
                 1D77     ;
                 1D77     ;        CTRLC - address of CTRL-C handling routine.
                 1D77     ;        CTRLZ - address of CTRL-Z handling routine.
                 1D77     ;
                 1D77     ; OUTPUTS:
                 1D77     ;
                 1D77     ;        None
                 1D77     ;
                 1D77     ; IMPLICIT OUTPUTS:
                 1D77     ;
                 1D77     ;        CTRL-C handler established for "CTRLC" routine.
                 1D77     ;        CTRL-Z handler established for "CTRLZ" routine.
                 1D77     ;        CTRLCZ_CHAN contains channel number.
                 1D77     ;
                 1D77     ; ROUTINE VALUE:
                 1D77     ;
                 1D77     ;        NORMAL, or failing system service status.
                 1D77     ;
                 1D77     ; SIDE EFFECTS:
                 1D77     ;
                 1D77     ;        none
                 1D77     ;
                 1D77     ;--
                 1D77
           001C  1D77     .ENTRY  ESTAB_CTRLCZ,   ^M<R2,R3,R4>
                 1D79
                 1D79             ALLOC   2,R1,R2                        ; Allocate word on stack for chan number
                 1D86             $ASSIGN_S DEVNAM=W^SYSCMD_DESC, CHAN=(R2) ; Assign channel to SYS$COMMAND
   03 50    E8   1D95             BLBS    R0,10$                         ; Continue if status OK
   008B     31   1D98             BRW     EC_ERR                         ; Branch if error
                 1D9B
                 1D9B     10$:
                 1D9B             ALLOC   DIB$K_LENGTH,R3,R4             ; Allocate DIB buffer on stack
                 1DB0             $GETCHN_S CHAN=(R2), PRIBUF=(R3) ; Get info on SYS$COMMAND device
   61 50    E9   1DC2             BLBC    R0,EC_ERR                      ; Branch if error
```

```
                               1DC5  6362
   04 A4    42 8F        91    1DC5  6363            CMPB    #DC$_TERM,DIB$B_DEVCLASS(R4) ; Is SYS$COMMAND device a terminal?
                  53     12    1DCA  6364            BNEQU   EC_NOR                   ; No -- go return with normal status
                               1DCC  6365
000000A1'EF    62        3C    1DCC  6366            MOVZWL  (R2),L^CTRLCZ_CHAN       ; Yes -- save channel no. for $CANCEL
                               1DD3  6367
                               1DD3  6368            $QIOW_S CHAN=(R2), -             ; Set up CTRL-C handler
                               1DD3  6369                    FUNC=#<IO$_SETMODE!IO$M_CTRLCAST>, -
                               1DD3  6370                    P1=G^CTRLC
                               1DF4  6371
            2F 50        E9    1DF4  6372            BLBC    R0,EC_ERR                ; Branch if error
                               1DF7  6373
                               1DF7  6374            $QIOW_S CHAN=(R2), -             ; Now set up CTRL-Z handler
                               1DF7  6375                    FUNC=#<IO$_SETMODE!IO$M_OUTBAND>, -
                               1DF7  6376                    P1=G^CTRLZ, -
                               1DF7  6377                    P2=#CTRLZ_MASK
                               1E1C  6378
            07 50        E9    1E1C  6379            BLBC    R0,EC_ERR                ; Branch if error
                               1E1F  6380
                               1E1F  6381 EC_NOR:
50    00000000'EF        D0    1E1F  6382            MOVL    NORMAL,R0                ; Normal status
                               1E26  6383 EC_ERR:
                         04    1E26  6384            RET                              ; Return with status
```

```
                              1E27          .SBTTL  ESTAB_CTRLW - Establish CTRL-W Handler
                              1E27
                              1E27    ;++
                              1E27    ;
                              1E27    ; FUNCTIONAL DESCRIPTION:
                              1E27    ;
                              1E27    ;       This routine is called to set up a CTRL-W handler for
                              1E27    ;       refreshing the display terminal screen. A channel is
                              1E27    ;       assigned to the display device and its device class is
                              1E27    ;       checked for TERMINAL. If not terminal class, the handler
                              1E27    ;       is not established.
                              1E27    ;
                              1E27    ;       Then a $QIOW is issued to the terminal driver to establish
                              1E27    ;       the CTRL-W handler. If any system service call fails,
                              1E27    ;       the failing status is returned; otherwise, NORMAL status
                              1E27    ;       is returned.
                              1E27    ;
                              1E27    ; INPUTS:
                              1E27    ;
                              1E27    ;       None
                              1E27    ;
                              1E27    ; IMPLICIT INPUTS:
                              1E27    ;
                              1E27    ;       MRBPTR - pointer to MRB (Monitor Request Block)
                              1E27    ;
                              1E27    ;       CTRLW - address of CTRL-W handling routine.
                              1E27    ;
                              1E27    ; OUTPUTS:
                              1E27    ;
                              1E27    ;       None
                              1E27    ;
                              1E27    ; IMPLICIT OUTPUTS:
                              1E27    ;
                              1E27    ;       CTRL-W handler established for ''CTRLW'' routine.
                              1E27    ;       CTRLW_CHAN contains channel number.
                              1E27    ;
                              1E27    ; ROUTINE VALUE:
                              1E27    ;
                              1E27    ;       NORMAL, or failing system service status.
                              1E27    ;
                              1E27    ; SIDE EFFECTS:
                              1E27    ;
                              1E27    ;       none
                              1E27    ;
                              1E27    ;--
                              1E27
                        001C  1E27    .ENTRY  ESTAB_CTRLW,    ^M<R2,R3,R4>
                              1E29
                              1E29            ALLOC   2,R1,R2                 ; Allocate word on stack for chan number
 53    00000000'EF      D0    1E36            MOVL    MRBPTR,R3               ; Get pointer to MRB
                              1E3D            $ASSIGN_S DEVNAM=@MRB$A_DISPLAY(R3), - ; Assign channel to display device
                              1E3D                      CHAN=(R2)
              67 50     E9    1E4B            BLBC    R0,EW_ERR               ; Branch if error
                              1E4E
                              1E4E            ALLOC   DIB$K_LENGTH,R3,R4      ; Allocate DIB buffer on stack
                              1E63            $GETCHN_S CHAN=(R2), PRIBUF=(R3) ; Get info on display device
              3D 50     E9    1E75            BLBC    R0,EW_ERR               ; Branch if error
```

```
                              1E78  6443
     04 A4    42 8F     91    1E78  6444            CMPB    #DC$_TERM,DIB$B_DEVCLASS(R4) ; Is display device a terminal?
                  2F    12    1E7D  6445            BNEQU   EW_NOR                       ; No -- go return with normal status
                              1E7F  6446
   000000A5'EF    62    3C    1E7F  6447            MOVZWL  (R2),L^CTRLW_CHAN            ; Yes -- save channel no. for $CANCEL
                              1E86  6448
                              1E86  6449            $QIOW_S CHAN=(R2), -                 ; Set up CTRL-W handler
                              1E86  6450                    FUNC=#<IO$_SETMODE!IO$M_OUTBAND>, -
                              1E86  6451                    P1=G^CTRLW, -
                              1E86  6452                    P2=#CTRLW_MASK
                              1EAB  6453
               07 50    E9    1EAB  6454            BLBC    R0,EW_ERR                    ; Branch if error
                              1EAE  6455
                              1EAE  6456 EW_NOR:
   50    00000000'EF    D0    1EAE  6457            MOVL    NORMAL,R0                    ; Normal status
                              1EB5  6458 EW_ERR:
                        04    1EB5  6459            RET                                  ; Return with status
```

```
                         1EB6         .SBTTL  MON_ERR - Log MONITOR Error
                         1EB6
                         1EB6   ;++
                         1EB6   ;
                         1EB6   ; FUNCTIONAL DESCRIPTION:
                         1EB6   ;
                         1EB6   ;       This routine is called to log an error whenever a MONITOR
                         1EB6   ;       synchronous error is discovered. Asynchronous (signaled)
                         1EB6   ;       errors are logged via the SIGNALED_ERR routine.
                         1EB6   ;       Logging consists of filling in the PUTMSGVEC array.
                         1EB6   ;       This array is the message argument vector for $PUTMSG which
                         1EB6   ;       will be called after all routines leading up to this
                         1EB6   ;       one have returned.
                         1EB6   ;
                         1EB6   ; INPUTS:
                         1EB6   ;
                         1EB6   ;        4(AP) - MONITOR message code (required parameter)
                         1EB6   ;
                         1EB6   ;        8(AP) - address of a secondary message code (0 if none).
                         1EB6   ;                (optional parameter)
                         1EB6   ;
                         1EB6   ;       12(AP) - 1st FAO argument for MONITOR message (optional
                         1EB6   ;                parameter). Up to 16 additional FAO arguments
                         1EB6   ;                may be included in this parameter list, immediately
                         1EB6   ;                following this parameter.
                         1EB6   ;
                         1EB6   ; IMPLICIT INPUTS:
                         1EB6   ;
                         1EB6   ;       PUTMSGVEC - 26-longword array, to contain the message argument
                         1EB6   ;                   vector for $PUTMSG.
                         1EB6   ;
                         1EB6   ; OUTPUTS:
                         1EB6   ;
                         1EB6   ;       none
                         1EB6   ;
                         1EB6   ; IMPLICIT OUTPUTS:
                         1EB6   ;
                         1EB6   ;       PUTMSGVEC contains message argument vector for later LIB$SIGNAL call.
                         1EB6   ;
                         1EB6   ; ROUTINE VALUE:
                         1EB6   ;
                         1EB6   ;       none
                         1EB6   ;
                         1EB6   ; SIDE EFFECTS:
                         1EB6   ;
                         1EB6   ;       none
                         1EB6   ;
                         1EB6   ;--
                         1EB6
                   000C  1EB6   .ENTRY  MON_ERR,        ^M<R2,R3>
                         1EB8
51   00002745'EF    DE   1EB8           MOVAL   PUTMSGVEC+4,R1          ; Get pointer to where MONITOR code goes
        81   04 AC  D0   1EBF           MOVL    4(AP),(R1)+             ; Move it in and point to next item in list
           52   6C  3C   1EC3           MOVZWL  (AP),R2                 ; Get number of input args
           52   01  D1   1EC6           CMPL    #1,R2                   ; Just one arg?
                09  19   1EC9           BLSS    10$                     ; No -- continue
00002741'EF     01  D0   1ECB           MOVL    #1,PUTMSGVEC            ; Yes -- tack on argument vector size
```

```
                     38   11  1ED2               BRB     ME_RET                        ; ... and go return
                              1ED4         10$:
                52   02   C2  1ED4               SUBL2   #2,R2                         ; Compute # of input FAO args
                     06   14  1ED7               BGTR    20$                           ; Continue if found some
                     81   D4  1ED9               CLRL    (R1)+                         ; Indicate none in PUTMSGVEC
                     52   D4  1EDB               CLRL    R2                            ; Remember for later
                     16   11  1EDD               BRB     50$                           ; ... and go check secondary code
                              1EDF         20$:
                52   16   D1  1EDF               CMPL    #PUTMSGSIZE-4,R2              ; # FAO args greater than max?
                     03   18  1EE2               BGEQ    30$                           ; No, OK as is
                52   16   D0  1EE4               MOVL    #PUTMSGSIZE-4,R2              ; Yes, replace with max
                              1EE7         30$:
                81   52   D0  1EE7               MOVL    R2,(R1)+                      ; Move # FAO args into list
                     53   D4  1EEA               CLRL    R3                            ; Clear an index register
                              1EEC         40$:
            81   0C AC43  D0  1EEC               MOVL    12(AP)[R3],(R1)+              ; Move an FAO arg into list
              F7 53    52  F2  1EF1              AOBLSS  R2,R3,40$                     ; Loop to move all FAO args
                              1EF5
                              1EF5         50$:
   00002741'EF  52   02   C1  1EF5               ADDL3   #2,R2,PUTMSGVEC               ; Compute # message args and store
                  08 AC   D5  1EFD               TSTL    8(AP)                         ; Secondary message code?
                     0A   13  1F00               BEQL    ME_RET                        ; No -- all done
              61   08 BC  D0  1F02               MOVL    @8(AP),(R1)                   ; Yes -- move in after FAO args
           00002741'EF    D6  1F06               INCL    PUTMSGVEC                     ; ... and count it
                              1F0C         ME_RET:
                          04  1F0C               RET                                   ; Return to caller
                          1F0D  6545         .SBTTL  SIGNALED_ERR - Log Signaled Error
                          1F0D  6546
                          1F0D  6547 ;++
                          1F0D  6548 ;
                          1F0D  6549 ; FUNCTIONAL DESCRIPTION:
                          1F0D  6550 ;
                          1F0D  6551 ;        This routine is called to log an error whenever a MONITOR
                          1F0D  6552 ;        asynchronous (signaled) error is discovered. Synchronous
                          1F0D  6553 ;        errors (detected by MONITOR) are logged via the MON_ERR
                          1F0D  6554 ;        routine. Logging consists of filling in the PUTMSGVEC array.
                          1F0D  6555 ;        This array is the message argument vector for $PUTMSG which
                          1F0D  6556 ;        will be called after all routines leading up to this
                          1F0D  6557 ;        one have returned.
                          1F0D  6558 ;
                          1F0D  6559 ; INPUTS:
                          1F0D  6560 ;
                          1F0D  6561 ;         4(AP) - MONITOR message code (required parameter)
                          1F0D  6562 ;
                          1F0D  6563 ;         8(AP) - secondary message code (required parameter)
                          1F0D  6564 ;
                          1F0D  6565 ;        12(AP) - number of additional (FAO) arguments for secondary
                          1F0D  6566 ;                 message.
                          1F0D  6567 ;
                          1F0D  6568 ;        16(AP) - address of first additional argument. Others
                          1F0D  6569 ;                 follow contiguously.
                          1F0D  6570 ;
                          1F0D  6571 ; IMPLICIT INPUTS:
                          1F0D  6572 ;
                          1F0D  6573 ;        PUTMSGVEC - 26-longword array, to contain the message argument
                          1F0D  6574 ;                    vector for $PUTMSG.
                          1F0D  6575 ;
                          1F0D  6576 ; OUTPUTS:
                          1F0D  6577 ;
                          1F0D  6578 ;        none
                          1F0D  6579 ;
                          1F0D  6580 ; IMPLICIT OUTPUTS:
                          1F0D  6581 ;
                          1F0D  6582 ;        PUTMSGVEC contains message argument vector for later LIB$SIGNAL call.
                          1F0D  6583 ;
                          1F0D  6584 ; ROUTINE VALUE:
                          1F0D  6585 ;
                          1F0D  6586 ;        none
                          1F0D  6587 ;
                          1F0D  6588 ; SIDE EFFECTS:
                          1F0D  6589 ;
                          1F0D  6590 ;        none
                          1F0D  6591 ;
                          1F0D  6592 ;--
                          1F0D  6593
                    000C  1F0D  6594 .ENTRY  SIGNALED_ERR,   ^M<R2,R3>
                          1F0F  6595
 51   00002745'EF   DE    1F0F  6596         MOVAL   PUTMSGVEC+4,R1          ; Get pointer to where MONITOR code goes
      81   04 AC    D0    1F16  6597         MOVL    4(AP),(R1)+             ; Move it in and point to next item in list
               81   D4    1F1A  6598         CLRL    (R1)+                   ; Zero MONITOR FAO args
      81   08 AC    D0    1F1C  6599         MOVL    8(AP),(R1)+             ; Move in secondary code
00002741'EF    03   D0    1F20  6600         MOVL    #3,PUTMSGVEC            ; Size of PUTMSGVEC so far
               52   D4    1F27  6601         CLRL    R2                      ; Start out with no PC/PSL args for 2ndary
```

```
          00   08 AC   0C   10   ED  1F29  6602          CMPZV   #STS$V_FAC_NO,#STS$S_FAC_NO,8(AP),#SYS_FAC_NO ; System fac code?
                            03   12  1F2F  6603          BNEQ    10$                        ; No -- go add in additional args
                       52   02   C0  1F31  6604          ADDL2   #2,R2                      ; Yes -- count the PC/PSL args (h'ware xcptn
                                     1F34  6605 10$:
                  52   0C   AC   C0  1F34  6606          ADDL2   12(AP),R2                  ; Add in caller's additional args for 2ndary
                            19   13  1F38  6607          BEQL    40$                        ; Go exit if none
                  53   10   AC   D0  1F3A  6608          MOVL    16(AP),R3                  ; Set up pointer to first add'l arg
                       52   16   D1  1F3E  6609          CMPL    #PUTMSGSIZE-4,R2           ; # FAO args greater than max?
                            03   18  1F41  6610          BGEQ    20$                        ; No, OK as is
                       52   16   D0  1F43  6611          MOVL    #PUTMSGSIZE-4,R2           ; Yes, replace with max
                                     1F46  6612 20$:
           00002741'EF      52   C0  1F46  6613          ADDL2   R2,PUTMSGVEC               ; Add the add'l args into PUTMSGVEC size
                                     1F4D  6614 30$:
                       81   83   D0  1F4D  6615          MOVL    (R3)+,(R1)+                ; Move from signal array to PUTMSGVEC
                         FA 52   F5  1F50  6616          SOBGTR  R2,30$                     ; Loop once for each add'l arg
                                     1F53  6617 40$:
                                 04  1F53  6618          RET                                ; ... and return
```

```
                                  1F54          .SBTTL  SIGNAL_MON_ERR - Signal MONITOR Error
                                  1F54
                                  1F54    ;++
                                  1F54    ;
                                  1F54    ; FUNCTIONAL DESCRIPTION:
                                  1F54    ;
                                  1F54    ;       This routine issues a CALLG to LIB$SIGNAL, passing a
                                  1F54    ;       signal argument list created by the MON_ERR or SIGNALED_ERR
                                  1F54    ;       routine. This routine is called from MONMAIN, a PL/I
                                  1F54    ;       routine; it is necessary because PL/I does not generate
                                  1F54    ;       G-form routine calls.
                                  1F54    ;
                                  1F54    ; INPUTS:
                                  1F54    ;
                                  1F54    ;       none
                                  1F54    ;
                                  1F54    ; IMPLICIT INPUTS:
                                  1F54    ;
                                  1F54    ;       PUTMSGVEC - 26-longword array, containing the signal argument
                                  1F54    ;                   list to be passed to LIB$SIGNAL.
                                  1F54    ;
                                  1F54    ; OUTPUTS:
                                  1F54    ;
                                  1F54    ;       none
                                  1F54    ;
                                  1F54    ; IMPLICIT OUTPUTS:
                                  1F54    ;
                                  1F54    ;       Condition is signaled. The VMS default condition handler will
                                  1F54    ;       display the error messages asssociated with the condition.
                                  1F54    ;
                                  1F54    ; ROUTINE VALUE:
                                  1F54    ;
                                  1F54    ;       none
                                  1F54    ;
                                  1F54    ; SIDE EFFECTS:
                                  1F54    ;
                                  1F54    ;       none
                                  1F54    ;
                                  1F54    ;--
                                  1F54
                            0000  1F54    .ENTRY  SIGNAL_MON_ERR, ^M<>
                                  1F56
00000000'GF    00002741'EF    FA  1F56          CALLG   L^PUTMSGVEC,G^LIB$SIGNAL ; Signal the MONITOR error
                              04  1F61          RET                              ; ... and return
```

```
                                        1F62         .SBTTL  LINK_MON_ERR - Link MONITOR Error
                                        1F62
                                        1F62  ;++
                                        1F62  ;
                                        1F62  ; FUNCTIONAL DESCRIPTION:
                                        1F62  ;
                                        1F62  ;        This routine is called to link a MONITOR error message
                                        1F62  ;        into PUTMSGVEC ahead of the message already there.
                                        1F62  ;        It uses two input arguments: the MONITOR error message
                                        1F62  ;        code and the address of its (only) argument.
                                        1F62  ;
                                        1F62  ; INPUTS:
                                        1F62  ;
                                        1F62  ;         4(AP) - MONITOR message code
                                        1F62  ;
                                        1F62  ;         8(AP) - address of the only FAO argument for the MONITOR
                                        1F62  ;                 message (must be present).
                                        1F62  ;
                                        1F62  ; IMPLICIT INPUTS:
                                        1F62  ;
                                        1F62  ;        PUTMSGVEC - 26-longword array, to contain the message argument
                                        1F62  ;                       vector for $PUTMSG.
                                        1F62  ;
                                        1F62  ; OUTPUTS:
                                        1F62  ;
                                        1F62  ;        none
                                        1F62  ;
                                        1F62  ; IMPLICIT OUTPUTS:
                                        1F62  ;
                                        1F62  ;        PUTMSGVEC updated to include the input MONITOR message as its
                                        1F62  ;        primary error, followed by the original contents of PUTMSGVFC
                                        1F62  ;        as a linked error.
                                        1F62  ;
                                        1F62  ; ROUTINE VALUE:
                                        1F62  ;
                                        1F62  ;        None
                                        1F62  ;
                                        1F62  ; SIDE EFFECTS:
                                        1F62  ;
                                        1F62  ;        None
                                        1F62  ;
                                        1F62  ;--
                                        1F62
                                  007C  1F62  .ENTRY  LINK_MON_ERR,   ^M<R2,R3,R4,R5,R6>
                                        1F64
                                        1F64         ALLOC   4*<PUTMSGSIZE-1>,R0,R6  ; Get temp space on stack
50   00002741'EF    04          C5      1F79         MULL3   #4,PUTMSGVEC,R0         ; Compute size of source for move
00   00002745'EF    50          2C      1F81         MOVC5   R0,PUTMSGVEC+4,#0, -    ; Move current contents to temp area
          66   0064 8F                  1F89
                                        1F8D                 #4*<PUTMSGSIZE-1>,(R6)
     00002741'EF    03          C0      1F8D         ADDL2   #3,PUTMSGVEC            ; Increase size of vector
   00002745'EF    04 AC         D0      1F94         MOVL    4(AP),PUTMSGVEC+4       ; Move error code into vector
     00002749'EF    01          D0      1F9C         MOVL    #1,PUTMSGVEC+8          ; Move FAO arg count into vector
   0000274D'EF    08 AC         D0      1FA3         MOVL    8(AP),PUTMSGVEC+12      ; Move FAO arg addr into vector
    00    66   0064 8F          2C      1FAB         MOVC5   #4*<PUTMSGSIZE-1>,(R6),#0 - ; Move orig contents back in
00002751'EF    0058 8F                  1FB1                 ,#4*<PUTMSGSIZE-4>,PUTMSGVEC+16
                                        1FB9
```

```
                              1FB9  6721          $PUTMSG_S MSGVEC=PUTMSGVEC      ; Put out the linked message
                  66   01  D0 1FCC  6722          MOVL    #1,(R6)                 ; Use temp area for another msg
       04 A6  00000000'8F  D0 1FCF  6723          MOVL    #MNR$_CONT,4(R6)        ; ... which says "Continuing..."
                              1FD7  6724
                              1FD7  6725          $PUTMSG_S MSGVEC=(R6)           ; Put it out
                           04 1FE6  6726          RET                             ; Return
```

```
                         1FE7          .SBTTL  FREE_MEM - Free Virtual Memory
                         1FE7
                         1FE7  ;++
                         1FE7  ;
                         1FE7  ; FUNCTIONAL DESCRIPTION:
                         1FE7  ;
                         1FE7  ;       This routine issues calls to LIB$FREE_VM to free up virtual
                         1FE7  ;       memory acquired by classes for FAO control strings and
                         1FE7  ;       collection buffer blocks. Also, a special write buffer used
                         1FE7  ;       by the PROCESSES class is freed if present; also, the SYSTEM
                         1FE7  ;       class DATA arrays. No status code checking is done, since this
                         1FE7  ;       routine is in a cleanup path.
                         1FE7  ;
                         1FE7  ; INPUTS:
                         1FE7  ;
                         1FE7  ;       None
                         1FE7  ;
                         1FE7  ; IMPLICIT INPUTS:
                         1FE7  ;
                         1FE7  ;       The CDB$L_FAOCTR, CDB$A_FAOCTR, CDB$L_BUFFERS and CDB$A_BUFFERS
                         1FE7  ;       fields contain the length and address, respectively, of memory
                         1FE7  ;       blocks to be freed (for each class in this MONITOR request).
                         1FE7  ;
                         1FE7  ;       Additionally, PROC_WRI_BUFD is a quadword containing the length
                         1FE7  ;       and address of the special write buffer for the PROCESSES class.
                         1FE7  ;
                         1FE7  ;       Also, SYS_DATA_ADDR and SYS_DATA_LEN describe the address and length
                         1FE7  ;       of the SYSTEM class DATA arrays.
                         1FE7  ;
                         1FE7  ; OUTPUTS:
                         1FE7  ;
                         1FE7  ;       None
                         1FE7  ;
                         1FE7  ; IMPLICIT OUTPUTS:
                         1FE7  ;
                         1FE7  ;       Memory is freed. Pointers to freed memory are cleared to 0.
                         1FE7  ;
                         1FE7  ; ROUTINE VALUE:
                         1FE7  ;
                         1FE7  ;       NORMAL
                         1FE7  ;
                         1FE7  ; SIDE EFFECTS:
                         1FE7  ;
                         1FE7  ;       None
                         1FE7  ;
                         1FE7  ;--
                         1FE7
                  00FC   1FE7  .ENTRY  FREE_MEM, ^M<R2,R3,R4,R5,R6,R7>
                         1FE9
                         1FE9  ;
                         1FE9  ; First free up memory left over from a special
                         1FE9  ; write buffer used for recording PRO'ESSES records.
                         1FE9  ;
                         1FE9
0000001E'EF         D5   1FE9          TSTL    L^PROC_WRI_BUFD+4       ; Is there a buffer?
         19         13   1FEF          BEQL    5$                      ; Br if not
0000001E'EF         DF   1FF1          PUSHAL  L^PROC_WRI_BUFD+4       ; Yes -- stack addr of buffer ptr
```

```
               0000001A'EF   DF  1FF7          PUSHAL  L^PROC_WRI_BUFD         ; Stack addr of buffer length
         00000000'GF    02   FB  1FFD          CALLS   #2,G^LIB$FREE_VM        ; Free the buffer
               0000001E'EF   D4  2004          CLRL    L^PROC_WRI_BUFD+4       ; Clear address
                                 200A
                                 200A  ;
                                 200A  ; Check for SYSTEM class DATA arrays, and free them if present
                                 200A  ;
                                 200A
                                 200A  5$:
               0000007B'EF   D5  200A          TSTL    SYS_DATA_ADDR           ; SYSTEM DATA arrays here ?
                        19   13  2010          BEQL    10$                     ; Branch if not
               0000007B'EF   DF  2012          PUSHAL  SYS_DATA_ADDR           ; Stack addr of arrays ptr
               0000007F'EF   DF  2018          PUSHAL  SYS_DATA_LEN            ; Stack addr of arrays length
         00000000'GF    02   FB  201E          CALLS   #2,G^LIB$FREE_VM        ; Free the space
               0000007B'EF   D4  2025          CLRL    SYS_DATA_ADDR           ; Clear address
                                 202B
                                 202B  ;
                                 202B  ; Now look only at the requested classes for this MONITOR request.
                                 202B  ; Free up the FAO control string and the collection buffer block for each.
                                 202B  ;
                                 202B
                                 202B  10$:
                                 202B
         57    00000000'EF   D0  202B          MOVL    MRBPTR,R7                       ; Load MRB pointer
                        55   D4  2032          CLRL    R5                              ; Init starting bit position
                                 2034  20$:
                  53    20   D0  2034          MOVL    #32,R3                          ; Init bit field size
                                 2037                                                  ; NOTE -- must handle in 32-bit chunks
                  52    55   D0  2037          MOVL    R5,R2                           ; Init start position of next chunk
                                 203A  30$:
   54    32 A7    53    52   EA  203A          FFS     R2,R3,MRB$O_CLASSBITS(R7),R4 ; Search for next class number
                                 2040                                                  ; R4 contains class no. if found
                        0E   13  2040          BEQL    40$                             ; Branch if none found this chunk
                        1C   10  2042          BSBB    FREE_CLASS                      ; Free memory for this class
                  53    52   C0  2044          ADDL2   R2,R3                           ; Compute next starting
           52     54    01   C1  2047          ADDL3   #1,R4,R2                        ; ... position and field size
                  53    52   C2  204B          SUBL2   R2,R3                           ; ... for this chunk
                        EA   11  204E          BRB     30$                             ; Go search rest of chunk
                                 2050  40$:
 FFDC 55    20    0000'8F    3D  2050          ACBW    #MAX_CLASS_NO,#32,R5,20$ ; Loop to process next chunk
                                 2058
         50    00000000'EF   D0  2058          MOVL    NORMAL,R0                       ; Set normal status
                             04  205F          RET                                     ; Return
                                 2060
                                 2060
                                 2060  FREE_CLASS:                                     ; Free class memory
                                 2060                                                  ; NOTE -- R4 contains class number
                                 2060
   56    54    00000053 8F   C5  2060          MULL3   #CDB$K_SIZE,R4,R6               ; Compute offset to desired CDB
      56     00000000'EF46   9E  2068          MOVAB   CDBHEAD[R6],R6                  ; Index to CDB address
                     04 A6   D5  2070          TSTL    CDB$A_FAOCTR(R6)                ; Is there an FAO control string?
                        23   13  2073          BEQL    10$                             ; Branch if not
         50    00000000'EF   DE  2075          MOVAL   SYS_FAO_STR,R0                  ; Get addr of special SYSTEM FAO str
               50    04 A6   D1  207C          CMPL    CDB$A_FAOCTR(R6),R0             ; Is this it?
                        13   13  2080          BEQL    5$                              ; Yes, don't try to free it
                     04 A6   DF  2082          PUSHAL  CDB$A_FAOCTR(R6)                ; Stack addr of string ptr
         66    000005DC 8F   D0  2085          MOVL    #FAOCTR_SIZE,CDB$L_FAOCTR(R6) ; Ensure whole string is freed
```

```
                 66   DF  208C  6842         PUSHAL  CDB$L_FAOCTR(R6)        ; Stack addr of string length
     00000000'GF  02   FB  208E  6843         CALLS   #2,G^LIB$FREE_VM        ; Free it
                           2095  6844 5$:
              04 A6   D4  2095  6845         CLRL    CDB$A_FAOCTR(R6)        ; Clear address
                           2098  6846 10$:
              2E A6   D5  2098  6847         TSTL    CDB$A_BUFFERS(R6)       ; Is there a buffer block?
                 10   13  209B  6848         BEQL    20$                     ; Branch if not
              2E A6   DF  209D  6849         PUSHAL  CDB$A_BUFFERS(R6)       ; Stack addr of block ptr
              2A A6   DF  20A0  6850         PUSHAL  CDB$L_BUFFERS(R6)       ; Stack addr of block length
     00000000'GF  02   FB  20A3  6851         CALLS   #2,G^LIB$FREE_VM        ; Free it
              2E A6   D4  20AA  6852         CLRL    CDB$A_BUFFERS(R6)       ; Clear address
                           20AD  6853 20$:
                      05  20AD  6854         RSB                             ; Return
```

```
20AE          .SBTTL  DISK_DISPNAM - DISK Class display name subroutine
20AE
20AE  ;++
20AE  ;
20AE  ; FUNCTIONAL DESCRIPTION:
20AE  ;
20AE  ;       This subroutine fills an FAO parameter stack with up to
20AE  ;       10 longwords required to display a single element (disk)
20AE  ;       name. The address of an element ID entry is passed to this
20AE  ;       routine in a register; the address of the FAO stack is
20AE  ;       also passed in a register.
20AE  ;
20AE  ; CALLING SEQUENCE:
20AE  ;
20AE  ;       JSB     DISK_DISPNAM
20AE  ;
20AE  ; INPUTS:
20AE  ;
20AE  ;       R3  = address of 10-longword FAO stack
20AE  ;       R6  = address of CDB
20AE  ;       R7  = address of CDX
20AE  ;       R10 = address of current element ID
20AE  ;
20AE  ; IMPLICIT INPUTS:
20AE  ;
20AE  ;       None
20AE  ;
20AE  ; OUTPUTS:
20AE  ;
20AE  ;       None
20AE  ;
20AE  ; IMPLICIT OUTPUTS:
20AE  ;
20AE  ;       The FAO parameter stack is filled with as many longword
20AE  ;       parameter values as necessary (up to 10) to display a
20AE  ;       single element name. The number of parameters is defined
20AE  ;       by the FAO control string defined for this homogeneous
20AE  ;       class.
20AE  ;
20AE  ; ROUTINE VALUE:
20AE  ;
20AE  ;       None
20AE  ;
20AE  ; SIDE EFFECTS:
20AE  ;
20AE  ;       Alters R0.
20AE  ;
20AE  ;--
```

```
                                20AE  6905 DISK_DISPNAM::
                                20AE  6906
                                20AE  6907 ;
                                20AE  6908 ; First determine whether we have the special wider name
                                20AE  6909 ; area available with the tabular (/ALL) statistics display.
                                20AE  6910 ;
                                20AE  6911
50   00000000'EF    D0          20AE  6912          MOVL    MRBPTR,R0                    ; Get MRB pointer
  0A 43 A0    0B    E0          20B5  6913          BBS     #MRB$V_MFSUM,MRB$W_FLAGS(R0),10$ ; Br if m.f. summary
     00    42 A6    91          20BA  6914          CMPB    CDB$B_ST(R6),#ALL_STAT       ; All statistics requested ?
              04    12          20BE  6915          BNEQ    10$                          ; Br if no
              50    D4          20C0  6916          CLRL    R0                           ; Indicate wide display area
              03    11          20C2  6917          BRB     20$                          ; ... and continue
                                20C4  6918 10$:
         50   01    D0          20C4  6919          MOVL    #1,R0                        ; Indicate narrow display area
                                20C7  6920 20$:
  32 4B A6    06    E1          20C7  6921          BBC     #CDB$V_DISKAC, -             ; Br if recorded without alloc class
                                20CC  6922                  CDB$L_FLAGS(R6),50$          ; .....
              6A    95          20CC  6923          TSTB    (R10)                        ; Allocation class 0?
              2E    13          20CE  6924          BEQL    50$                          ; Br if so
                                20D0  6925
                                20D0  6926 ;
                                20D0  6927 ; Process a disk name with allocation class
                                20D0  6928 ;
                                20D0  6929
 2C A7    25BA'CF   DE          20D0  6930          MOVAL   W^DISK_FAO_AC, -             ; FAO string with alloc class
                                20D6  6931                  CDX$A_DISPFAO(R7)
         63   6A    9A          20D6  6932          MOVZBL  (R10),(R3)                   ; Move alloc class into FAO stack
  04 A3    01 AA    DE          20D9  6933          MOVAL   1(R10),4(R3)                 ; Move in device name pointer
  08 A3    05 AA    3C          20DE  6934          MOVZWL  5(R10),8(R3)                 ; ... and unit number
                                20E3  6935
           0C A3    D4          20E3  6936          CLRL    12(R3)                       ; Assume wide display
           04 50    E9          20E6  6937          BLBC    R0,30$                       ; Br if so
     0C A3    01    D0          20E9  6938          MOVL    #1,12(R3)                    ; Else stack a different value
                                20ED  6939 30$:
           10 A3    D4          20ED  6940          CLRL    16(R3)                       ; Assume zero length node name field
           04 50    E8          20F0  6941          BLBS    R0,40$                       ; Br if narrow display area
     10 A3    09    D0          20F3  6942          MOVL    #9,16(R3)                    ; Stack len of node name field for wide
                                20F7  6943 40$:
  14 A3    07 AA    7E          20F7  6944          MOVAQ   7(R10),20(R3)                ; Stack node name address
              51    11          20FC  6945          BRB     DD_VOL                       ; ... and go stack volume name
                                20FE  6946
                                20FE  6947 ;
                                20FE  6948 ; Process a disk name with the node$device format.
                                20FE  6949 ;
                                20FE  6950
                                20FE  6951 50$:
 2C A7    2598'CF   DE          20FE  6952          MOVAL   W^DISK_FAO,CDX$A_DISPFAO(R7) ; FAO string without alloc class
         63   0D    D0          2104  6953          MOVL    #13,(R3)                     ; Assume narrow display area
     14 A3    01    D0          2107  6954          MOVL    #1,20(R3)                    ; .......
           06 50    E8          210B  6955          BLBS    R0,60$                       ; Br if narrow
         63   16    D0          210E  6956          MOVL    #22,(R3)                     ; Stack values for wide area
           14 A3    D4          2111  6957          CLRL    20(R3)                       ; .......
                                2114  6958 60$:
  1D 4B A6    06    E1          2114  6959          BBC     #CDB$V_DISKAC, -             ; Br if recorded without alloc class
                                2119  6960                  CDB$L_FLAGS(R6),80$          ; ....
                                2119  6961
```

```
                          2119  6962 ;
                          2119  6963 ; Process a disk name with with zero allocation class
                          2119  6964 ;
                          2119  6965
04 A3    07 AA     7E     2119  6966         MOVAQ   7(R10),4(R3)          ; Move node name ptr into FAO stack
         08 A3     D4     211E  6967         CLRL    8(R3)                 ; Assume zero dollar-sign field length
         07 AA     95     2121  6968         TSTB    7(R10)                ; See if node name exists
            04     13     2124  6969         BEQL    70$                   ; Br if not
   08 A3    01     D0     2126  6970         MOVL    #1,8(R3)              ; Adjust dollar-sign field length
                          212A  6971 70$:
0C A3    01 AA     DE     212A  6972         MOVAL   1(R10),12(R3)         ; Move in device name pointer
10 A3    05 AA     3C     212F  6973         MOVZWL  5(R10),16(R3)         ; ... and unit number
            19     11     2134  6974         BRB     DD_VOL                ; Go stack volume name
```

```
                                2136  6976 ;
                                2136  6977 ; Process a disk name recorded without allocation class.
                                2136  6978 ; This is the revision level 0 format for the element ID.
                                2136  6979 ; It does not include an allocation class, and the fields
                                2136  6980 ; are in positions different from those of later revision
                                2136  6981 ; levels. This code is also suitable for processing journal
                                2136  6982 ; device names.
                                2136  6983 ;
                                2136  6984
                                2136  6985 80$:
           04 A3    6A    7E    2136  6986          MOVAQ   (R10),4(R3)             ; Move node name ptr into FAO stack
              08 A3       D4    213A  6987          CLRL    8(R3)                   ; Assume zero dollar-sign field length
                    6A    95    213D  6988          TSTB    (R10)                   ; See if node name exists
                    04    13    213F  6989          BEQL    90$                     ; Br if not
           08 A3    01    D0    2141  6990          MOVL    #1,8(R3)                ; Adjust dollar-sign field length
                                2145  6991 90$:
        0C A3    08 AA    DE    2145  6992          MOVAL   8(R10),12(R3)           ; Move in device name pointer
        10 A3    0C AA    3C    214A  6993          MOVZWL  12(R10),16(R3)          ; ... and unit number
                                214F  6994
                                214F  6995 DD_VOL:
              18 A3       D4    214F  6996          CLRL    24(R3)                  ; Assume no volume name
        1C A3    FE AF    D0    2152  6997          MOVL    .,28(R3)                ; .... (use any accessible address)
        09 4B A6    07    E1    2157  6998          BBC     #CDB$V_DISKVN, -        ; Br if volume name not available
                                215C  6999                  CDB$L_FLAGS(R6),DD_RSB  ; ....
           18 A3    0C    D0    215C  7000          MOVL    #12,24(R3)              ; Stack length of vol name
        1C A3    0F AA    9E    2160  7001          MOVAB   15(R10),28(R3)          ; ... and its address
                                2165  7002
                                2165  7003 DD_RSB:
                          05    2165  7004          RSB                             ; Return to caller
                                2166  7005
```

```
                                 2166          .SBTTL  SCS_DISPNAM - SCS Class display name subroutine
                                 2166
                                 2166   ;++
                                 2166   ;
                                 2166   ; FUNCTIONAL DESCRIPTION:
                                 2166   ;
                                 2166   ;      This subroutine fills an FAO parameter stack with up to
                                 2166   ;      10 longwords required to display a single element (SCS)
                                 2166   ;      name. The address of an element ID entry is passed to this
                                 2166   ;      routine in a register; the address of the FAO stack is
                                 2166   ;      also passed in a register.
                                 2166   ;
                                 2166   ; CALLING SEQUENCE:
                                 2166   ;
                                 2166   ;      JSB     SCS_DISPNAM
                                 2166   ;
                                 2166   ; INPUTS:
                                 2166   ;
                                 2166   ;      R3  = address of 10-longword FAO stack
                                 2166   ;      R10 = address of current element ID
                                 2166   ;
                                 2166   ; IMPLICIT INPUTS:
                                 2166   ;
                                 2166   ;      None
                                 2166   ;
                                 2166   ; OUTPUTS:
                                 2166   ;
                                 2166   ;      None
                                 2166   ;
                                 2166   ; IMPLICIT OUTPUTS:
                                 2166   ;
                                 2166   ;      The FAO parameter stack is filled with as many longword
                                 2166   ;      parameter values as necessary (up to 10) to display a
                                 2166   ;      single element name. The number of parameters is defined
                                 2166   ;      by the FAO control string defined for this homogeneous
                                 2166   ;      class.
                                 2166   ;
                                 2166   ; ROUTINE VALUE:
                                 2166   ;
                                 2166   ;      None
                                 2166   ;
                                 2166   ; SIDE EFFECTS:
                                 2166   ;
                                 2166   ;      None
                                 2166   ;
                                 2166   ;--
                                 2166
                                 2166   SCS_DISPNAM::
                                 2166
         63       6A     7E      2166          MOVAQ   (R10),(R3)             ; Move node name ptr into FAO stack
                  6A     95      2169          TSTB    (R10)                  ; Is there a node name?
                  07     12      216B          BNEQ    10$                    ; Yes, return
 63     000025F1'EF      9E      216D          MOVAB   UNKNOWN_NODE,(R3)      ; No, make it 'Unknown Node''
                                 2174   10$:
                         05      2174          RSB                            ; Return to caller
                                 2175
                                 2175   .END
```

```
$$T1                            = 00000001                    CDB$L_FLAGS                   = 0000004B
ACT_TO_CUR                      = 00000002  G                 CDB$L_ICOUNT                  = 00000014
ADV_HOM_ITEM                      000006C7 RG     03          CDB$L_MIN                     = 00000038
ALL_STAT                        = 00000000                    CDB$L_RANGE                   = 0000003C
ALL_TO_ACT                      = 00000003  G                 CDB$L_SUMBUF                  = 00000008
ANNCE_STR                         000022DD RG     01          CDB$M_CPU                     = 00000002
ATTRIBMSK                         0000272B R      01          CDB$M_CPU_COMB                = 00000008
AVE_STAT                        = 00000002                    CDB$M_CTPRES                  = 00000001
BAR1                              ********   X    03          CDB$M_DISABLE                 = 00000200
BAR10                             ********   X    03          CDB$M_DISKAC                  = 00000040
BAR11                             ********   X    03          CDB$M_DISKVN                  = 00000080
BAR2                              ********   X    03          CDB$M_EXPLIC                  = 00001000
BAR3                              ********   X    03          CDB$M_HOMOG                   = 00000020
BAR4                              ********   X    03          CDB$M_KUNITS                  = 00000400
BAR5                              ********   X    03          CDB$M_PERCENT                 = 00000001
BAR6                              ********   X    03          CDB$M_STD                     = 00000010
BAR7                              ********   X    03          CDB$M_SWAPBUF                 = 00000002
BAR8                              ********   X    03          CDB$M_SYSCLS                  = 00000100
BAR9                              ********   X    03          CDB$M_UNIFORM                 = 00000004
BARCHAR                           00000005 RG     01          CDB$M_WIDE                    = 00000800
BARHEAD_STR                       000024DD RG     01          CDB$S_CDB                     = 00000053
BARSIZE                           00000001 RG     01          CDB$S_FILLER                  = 00000013
BAR_LWORDS                      = 00000003                    CDB$S_FLAGS                   = 00000004
BET_EV_FLAG                       ********   X    03          CDB$S_QFILLER                 = 0000000E
BET_SCREENS                       000000A9 R      01          CDB$S_QFLAGS                  = 00000002
BLANK_STR                         000023F5 RG     01          CDB$V_CPU                     = 00000001
BLINK                             000016FF R      03          CDB$V_CPU_COMB                = 00000003
BOT_CURS                          000022D6 RG     01          CDB$V_CTPRES                  = 00000000
BPU                               00000006 R      01          CDB$V_DISABLE                 = 00000009
BS_SECS                         = 00000002                    CDB$V_DISKAC                  = 00000006
BU_SYS_SINGLE                     ********   X    03          CDB$V_DISKVN                  = 00000007
CALC_BAR                          00000F2E R      03          CDB$V_EXPLIC                  = 0000000C
CALC_CLASS                        0000004B R      03          CDB$V_FILLER                  = 0000000D
CALC_DITEM                        000000D2 R      03          CDB$V_HOMOG                   = 00000005
CALC_LEN                          00000000 RG     03          CDB$V_KUNITS                  = 0000000A
CB_ADDRS                          000000D8 RG     01          CDB$V_PERCENT                 = 00000000
CC_ERROR                          000003EF R      03          CDB$V_QFILLER                 = 00000002
CC_NORMAL                         000003E8 R      03          CDB$V_STD                     = 00000004
CDB                             = 00000000                    CDB$V_SWAPBUF                 = 00000001
CDB$A_BUFFERS                   = 0000002E                    CDB$V_SYSCLS                  = 00000008
CDB$A_CDX                       = 00000032                    CDB$V_UNIFORM                 = 00000002
CDB$A_CHDHDR                    = 0000004F                    CDB$V_WIDE                    = 0000000B
CDB$A_FAOCTR                    = 00000004                    CDB$W_BLKLEN                  = 00000020
CDB$A_ITMSTR                    = 0000001C                    CDB$W_DISPCTL                 = 00000036
CDB$A_POSTCOLL                  = 00000026                    CDB$W_QFLAGS                  = 00000045
CDB$A_PRECOLL                   = 00000022                    CDB$W_QFLAGS_CUR              = 00000049
CDB$A_SUMBUF                    = 0000000C                    CDB$W_QFLAGS_DEF              = 00000047
CDB$A_TITLE                     = 00000010                    CDBHEAD                         ********   X    03
CDB$B_FAOPRELEN                 = 00000041                    CDBPTR                          ********   X    03
CDB$B_FAOSEGLEN                 = 00000040                    CDB_EXT                       = 00000000
CDB$B_ST                        = 00000042                    CDX$A_DISPFAO                 = 0000002C
CDB$B_ST_CUR                    = 00000044                    CDX$A_DISPNAM                 = 00000028
CDB$B_ST_DEF                    = 00000043                    CDX$A_ELIDTABLE               = 0000000C
CDB$K_SIZE                      = 00000053                    CDX$A_ILOOKTAB                = 00000024
CDB$L_BUFFERS                   = 0000002A                    CDX$A_SCBTABLE                = 00000010
CDB$L_ECOUNT                    = 000000'8                    CDX$A_SELIDTABLE              = 00000018
CDB$L_FAOCTR                    = 00000000                    CDX$B_ELIDLEN                 = 00000009
```

```
CDX$B_IDISCONSEC                 = 00000007                  CURXPOS                            00000012 R      01
CDX$B_IDISCT                     = 00000006                  CUR_STAT                         = 00000001
CDX$B_IDISINDEX                  = 00000008                  CUR_TO_ACT                       = 00000001  G
CDX$K_SIZE                       = 00000030                  CVT_TO_DELTA                       00000587 RG     03
CDX$L_DCOUNT                     = 0000001C                  DC$_TERM                         = 00000042
CDX$L_PREV_DCT                   = 00000020                  DD_RSB                             00002165 R      03
CDX$L_SELIDTABLE                 = 00000014                  DD_VOL                             0000214F R      03
CDX$S_CDB_EXT                    = 00000030                  DEC_CRT                          = 00000000
CDX$S_IBITS                      = 00000010                  DEF$A_DISP                       = 0000000C
CDX$W_CUMELCT                    = 0000000A                  DEF$A_REC                        = 00000004
CDX$W_IBITS                      = 000C0000                  DEF$A_SUMM                       = 00000014
CDX$W_IBITS_CUR                  = 00000004                  DEF$L_DISP                       = 00000008
CDX$W_IBITS_DEF                  = 00000002                  DEF$L_REC                        = 00000000
CHD                              = 00000000                  DEF$L_SUMM                       = 00000010
CHD$A_ITMSTR                     = 00000004                  DEF$S_DEF_DESC                   = 00000018
CHD$B_ELIDLEN                    = 0000000A                  DEF_BAR                          = 0000002A
CHD$K_SIZE                       = 0000000D                  DEF_DESC                         = 00000000
CHD$L_ICOUNT                     = 00000000                  DEF_TO_CUR                       = 00000000  G
CHD$S_CHD                        = 0000000D                  DHN_RET                            00001AD9 R      03
CHD$W_BLKLEN                     = 00000008                  DHOMOG_ERR                         000018E0 R      03
CHD$W_DISPCTL                    = 0000000B                  DHOMOG_RET                         000018DF R      03
CHEKBUF                            00001C6C R      03        DIB$B_DEVCLASS                   = 00000004
CHEKRET                            00001C62 R      03        DIB$K_LENGTH                     = 00000074
CLASS_COLLECT                      00000258 RG     03        DISK_DISPNAM                       000020AE RG     03
CLASS_HDR                        = 00000000                  DISK_FAO                           00002598 R      01
CLEAR_DATA                         ********    X   03        DISK_FAO_AC                        000025BA R      01
CLEAR_STACK                        00001198 R      03        DISPERR                            00001A2C R      03
CLRVT55                            000022B7 RG     01        DISPLAY_HOMOG                      0000180E RG     03
CLU$GL_CLUB                        ********    X   03        DISPLAY_INIT                       000009A0 RG     03
CLUS_NET_INFO                      000005D7 RG     03        DISPLAY_PROCS                      00001511 RG     03
CNI_RET                            000006C6 R      03        DISPLAY_PUT                        00001ADA RG     03
CNI_SUCC                           000006BF R      03        DISPLAY_TOP                        00001788 RG     03
COLLECTION                         00000700 R      03        DISP_HOM_ITMNAM                    000018E4 R      03
COLLECTION_END                     ********    X   03        DISP_HOM_NAMES                     00001A48 RG     03
COLL_BUFS                        = 00000002  G               DPROCS_ERR                         00001689 R      03
COLL_COMM                          00000764 R      03        DPROCS_RET                         00001688 R      03
COLL_NONSTD                        0000078D R      03        DP_ERR                             00001B59 R      03
COLL_RSB                           0000078C R      03        DSC$K_CLASS_D                    = 00000002
COMBINE_MODES                      00000423 R      03        DSC$K_DTYPE_T                    = 0000000E
COMMON                             00000E5C R      03        DT$_VT100                        = 00000060
COMMON_INIT                        00000AA8 R      03        DT$_VT52                         = 00000040
COMM_STR                           00002341 RG     01        DT$_VT55                         = 00000041
COMPUTE_BOOTTIME                   00000595 RG     03        DTOP_RET                           0000180D R      03
COMPUTE_STATS                      000008DF R      03        DYN_STRING                         000000FB RG     01
COUNT_TYPE                         ********    X   03        ECOUNT_SYS_ALL                     ********    X   03
CPU_BUSY                           ********    X   03        ECOUNT_SYS_SINGLE                  ********    X   03
CR                               = 0000000D                  EC_ERR                             00001E26 R      03
CTRLC                              ********    X   03        EC_NOR                             00001E1F R      03
CTRLCZ_CHAN                        000000A1 RG     01        ERLINE_STR                         000026EF R      01
CTRLCZ_HIT                         ********    X   03        ERLNNO                           = 000026F2 R      01
CTRLW                              ********    X   03        ESC                              = 0000001B
CTRLW_CHAN                         000000A5 RG     01        ESC_SEQ_TABLE                      00002737 R      01
CTRLW_MASK                         00000091 R      01        ESTAB_CTRLCZ                       00001D77 RG     03
CTRLZ                              ********    X   03        ESTAB_CTRLW                        00001E27 RG     03
CTRLZ_MASK                         00000099 R      01        ES_TAB_LEN                       = 0000000A
CURGRAPH                           0000000E RG     01        EW_ERR                             00001EB5 R      03
CURSOR_STR                         00002540 RG     01        EW_NOR                             00001EAE R      03
```

```
EXE$GB_CPUTYPE                  ********   X   03    IDB$A_ADDR                    = 0000000C
EXE$GL_ABSTIM                   ********   X   03    IDB$A_LNAME                   = 00000004
EXE$GL_RPB                      ********   X   03    IDB$A_SNAME                   = 00000000
EXE$GQ_SYSTIME                  ********   X   03    IDB$B_FLAGS                   = 00000010
FACNO                         = 000000CE             IDB$K_ILENGTH                 = 00000011
FAOCTR_SIZE                   = 000005DC  G          IDB$M_PCNT                    = 00000001
FAOSTK                          00000103 RG    01    IDB$S_FILLER                  = 00000007
FAOSTK_SIZE                   = 00001900             IDB$S_FLAGS                   = 00000001
FDB_BEG                         00000DBE R     03    IDB$S_IDB                     = 00000011
FDB_REGPROC                     00000C10 R     03    IDB$V_FILLER                  = 00000001
FDB_RET                         00000E6B R     03    IDB$V_PCNT                    = 00000000
FDB_STD                         00000CD3 R     03    IDB$W_ISIZE                   = 00000008
FDB_SYS_ALL                     00001209 R     03    IDB$W_TYPE                    = 0000000A
FDB_SYS_SINGLE                  00000FA4 R     03    ILN_REG                       = 00000006
FDB_SYS_TOP                     00001089 RG    03    INTAVE                          00000E86 R     03
FETCH                           000001DF RG    03    INTORFL                         00000EF3 R     03
FHS_ERR                         00001A28 R     03    IO$M_CTRLCAST                 = 00000100
FHS_RET                         00001A27 R     03    IO$M_OUTBAND                  = 00000400
FILE_HDR                      = 00000000             IO$_SETMODE                   = 00000023
FILL_DISP_BUFF                  00000B87 RG    03    ITEM_NAM_STR                    000025FE R     01
FILL_HETERO_STATS               000007E9 R     03    ITEM_TYPE                       000000E0 R     01
FILL_HOMOG_SCREEN               00001934 RG    03    ITMLNNO                       = 00002601 R     01
FILL_HOMOG_STATS                ********   X   03    ITMSTR_SYS_ALL                  ********   X   03
FILL_PCSTATS_BUFF               000008A8 R     03    ITMSTR_SYS_SINGLE               ********   X   03
FILL_SCREEN                     000016A5 R     03    K_STR                           0000256E RG    01
FILL_TOP                        000012A7 RG    03    LARGE_NO                      = 7FFF7FFF  G
FIND_TOP                        000011A7 R     03    LAST_DATA_LINE                = 00000016  G
FIN_SEQ                         000022C9 RG    01    LF                            = 0000000A
FIRST_DATA_LINE               = 00000008  G          LIB$ERASE_LINE                  ********   X   03
FMT_SYS_SINGLE                  ********   X   03    LIB$ERASE_PAGE                  ********   X   03
FOOTP                           000023F7 R     01    LIB$FREE_VM                     ********   X   03
FOOTR                           000023FF R     01    LIB$GET_VM                      ********   X   03
FOOTS                           000023FB R     01    LIB$PUT_BUFFER                  ********   X   03
FREE_CLASS                      00002060 R     03    LIB$SET_BUFFER                  ********   X   03
FREE_MEM                        00001FE7 RG    03    LIB$SIGNAL                      ********   X   03
FSS_BEG                         00001056 R     03    LINK_MON_ERR                    00001F62 RG    03
FS_ALL                          00000FE0 R     03    MAX55HEIGHT                   = 000000C8
FS_AVE                          00000FEE R     03    MAXBARS                       = 00000028  G
FS_COMMON                       00001018 R     03    MAXBARS_SYS                   = 0000001A  G
FS_CUR                          00000FE0 R     03    MAXELTS                       = 000000C8  G
FS_MAX                          0000100A R     03    MAXELTS_MFS                   = 00000190  G
FS_MIN                          00000FFC R     03    MAX_CLASS_NO                    ********   X   03
FT_CASE                         000012D3 R     03    MAX_ELIDLEN                   = 0000001B  G
F_ALL                           00000DE3 R     03    MAX_HOM_ITEMS                 = 0000000F  G
F_AVE                           00000DFF R     03    MAX_REC_SIZE                  = 00007D00  G
F_CUR                           00000DE3 R     03    MAX_STAT                      = 00000004
F_MAX                           00000E44 R     03    MBP                           = 00000000
F_MIN                           00000E28 R     03    MBP$A_ADDR                    = 00000018
GET_BUFFERS                     ********   X   03    MBP$A_B1ST                    = 00000004
GET_COMPUTED_ITEMS              0000087D R     03    MBP$A_BA                      = 00000000
GMIN                            0000000A R     01    MBP$A_BUFF1ST                 = 00000004
HARDCOPY                      = 00000002             MBP$A_BUFFA                   = 00000000
HOLD_SCREEN                     0000174C R     03    MBP$A_BUFFERA                 = 00000000
HOMOG_TYPE                      000000E2 R     01    MBP$A_BUFFERB                 = 00000004
HOM_CLASS_PRE                 = 00000000             MBP$A_DATA                    = 00000008
HORIZ_STR                       00002542 RG    01    MBP$A_DIFF                    = 0000000C
IDB                           = 00000000             MBP$A_MAX                     = 00000010
```

```
MBP$A_MIN                      = 0000000C                 MFS$B_DATA_COLS                = 0000003C
MBP$A_ORDER                    = 00000010                 MFS$K_SIZE                     = 0000003D
MBP$A_PCMAX                    = 00000020                 MFS$L_ELEMS                    = 00000030
MBP$A_PCMIN                    = 0000001C                 MFS$L_LWORDS                   = 00000034
MBP$A_PCSTATS                  = 00000018                 MFS$L_STATSBUF                 = 00000020
MBP$A_PCSUM                    = 00000024                 MFS$O_CLASSBITS                = 00000000
MBP$A_PID                      = 00000014                 MFS$Q_BEGINNING                = 00000010
MBP$A_PR_FAOSTK                = 00000008                 MFS$Q_ENDING                   = 00000018
MBP$A_STATS                    = 00000008                 MFS$S_BEGINNING                = 00000008
MBP$A_SUM                      = 00000014                 MFS$S_CLASSBITS                = 00000010
MBP$K_SIZE                     = 00000028                 MFS$S_ENDING                   = 00000008
MBP$S_MBP                      = 00000028                 MFS$S_MFS                      = 0000003D
MBP$S_MBP2                     = 0000001C                 MFS$W_CLASSCT                  = 00000038
MBP$S_MBP3                     = 0000000C                 MFSPTR                           ********   X   03
MBP2                           = 00000000                 MIN_STAT                       = 00000003
MBP3                           = 00000000                 MNR$_CONT                        ********   X   03
MCA                            = 00000000                 MNR$_DISPERR                     ********   X   03
MCA$A_INPUT_PTR                = 00000004                 MNR$_ITMNOTDEF                   ********   X   03
MCA$A_MPADDR                   = 0000001C                 MNR$_OPENIN                    = 00CE109A  G
MCA$B_FIRSTC                   = 00000030                 MNR$_PREMEOF                     ********   X   03
MCA$B_LASTC                    = 00000031                 MNR_CLS$B_TYPE                 = 00000000
MCA$K_SIZE                     = 0000003A                 MNR_CLS$K_HSIZE                = 0000000D
MCA$L_COLLCNT                  = 0000000C                 MNR_CLS$Q_STAMP                = 00000003
MCA$L_CONSEC_REC               = 00000034                 MNR_CLS$S_CLASS_HDR            = 0000000D
MCA$L_DISPCNT                  = 00000010                 MNR_CLS$S_FILLER               = 0000000F
MCA$L_INPUT_LEN                = 00000000                 MNR_CLS$S_FLAGS                = 00000002
MCA$L_INTTICKS                 = 00000008                 MNR_CLS$S_STAMP                = 00000008
MCA$L_INT_MULT                 = 00000014                 MNR_CLS$V_CONT                 = 00000000
MCA$L_PROC_DISP                = 00000018                 MNR_CLS$V_FILLER               = 00000001
MCA$Q_CURR_TIME                = 00000020                 MNR_CLS$W_FLAGS                = 00000001
MCA$Q_LASTCOLL                 = 00000028                 MNR_CLS$W_RESERVED             = 0000000B
MCA$S_CURR_TIME                = 00000008                 MNR_HDR$B_TYPE                 = 00000000
MCA$S_FILLER                   = 00000006                 MNR_HDR$K_CLASSBITS            = 00000073
MCA$S_FLAGS                    = 00000002                 MNR_HDR$K_MAXCOMLEN            = 0000003C
MCA$S_LASTCOLL                 = 00000008                 MNR_HDR$K_REVLEVELS            = 00000083
MCA$S_MCA                      = 0000003A                 MNR_HDR$K_SIZE                 = 00000103
MCA$V_ENTRY                    = 00000000                 MNR_HDR$L_FLAGS                = 00000001
MCA$V_EOF                      = 00000003                 MNR_HDR$L_INTERVAL             = 00000015
MCA$V_ERA_SCRL                 = 00000006                 MNR_HDR$L_RECCT                = 00000029
MCA$V_FILLER                   = 0000000A                 MNR_HDR$O_CLASSBITS            = 00000073
MCA$V_FUTURE                   = 00000001                 MNR_HDR$O_REVOCLSBITS          = 00000019
MCA$V_GRAPHICS                 = 00000005                 MNR_HDR$Q_BEGINNING            = 00000005
MCA$V_MULTFND                  = 00000002                 MNR_HDR$Q_ENDING               = 0000000D
MCA$V_REFRESH                  = 00000008                 MNR_HDR$S_BEGINNING            = 00000008
MCA$V_S_TOP_DISP               = 00000009                 MNR_HDR$S_CLASSBITS            = 00000010
MCA$V_TOP_DISP                 = 00000007                 MNR_HDR$S_COMMENT              = 0000003C
MCA$V_VIDEO                    = 00000004                 MNR_HDR$S_ENDING               = 00000008
MCA$W_DCLASSCT                 = 00000038                 MNR_HDR$S_FILE_HDR             = 00000103
MCA$W_FLAGS                    = 00000032                 MNR_HDR$S_FILLER               = 00000020
MCAPTR                           ********   X   03        MNR_HDR$S_FLAGS                = 00000004
ME_RET                           00001F0C R     03        MNR_HDR$S_LEVEL                = 00000008
MFS                            = 00000000                 MNR_HDR$S_REVOCLSBITS          = 00000010
MFS$A_IFB_TAB                  = 00000028                 MNR_HDR$S_REVLEVELS            = 00000080
MFS$A_STATSBUF                 = 00000024                 MNR_HDR$S_TYPE                 = 00000008
MFS$A_SUMMARY                  = 0000002C                 MNR_HDR$T_COMMENT              = 00000035
MFS$B_COLUMNS                  = 0000003A                 MNR_HDR$T_LEVEL                = 0000002D
MFS$B_CUR_COL                  = 0000003B                 MNR_HDR$T_REVLEVELS            = 00000083
```

```
MNR_HDR$V_FILLER                = 00000000              MOVE_CHD                         00001D49 R      03
MNR_HDR$W_COMLEN                = 00000071              MOVE_CLASS_QUALS                 0000012B RG     03
MNR_HOM$K_PSIZE                 = 00000008              MOVE_ITEMS                       000012A0 R      03
MNR_HOM$L_ELTCT                 = 00000000              MOVE_TOP8                        00001383 R      03
MNR_HOM$L_RESERVED              = 00000004              MPCHECK                          0000046A RG     03
MNR_HOM$S_HOM_CLASS_PRE         = 00000008              MRB                            = 00000000
MNR_PRO$B_PRI                   = 0000000A              MRB$A_COMMENT                  = 0000002C
MNR_PRO$K_DSIZE                 = 0000003B              MRB$A_DISPLAY                  = 00000020
MNR_PRO$K_FSIZE                 = 00000040              MRB$A_INPUT                    = 0000001C
MNR_PRO$K_PSIZE                 = 00000008              MRB$A_RECORD                   = 00000024
MNR_PRO$K_REV0DSIZE             = 00000033              MRB$A_SUMMARY                  = 00000028
MNR_PRO$K_REV1DSIZE             = 0000003B              MRB$B_INP_FILES                = 00000042
MNR_PRO$L_BIOCNT                = 0000002F              MRB$K_SIZE                     = 00000045
MNR_PRO$L_CPUTIM                = 0000002B              MRB$L_FLUSH                    = 00000014
MNR_PRO$L_DIOCNT                = 00000023              MRB$L_INTERVAL                 = 00000010
MNR_PRO$L_EFWM                  = 00000037              MRB$L_VIEWING_TIME             = 00000018
MNR_PRO$L_EPID                  = 00000033              MRB$M_ALL_CLASS                = 00000400
MNR_PRO$L_IPID                  = 00000000              MRB$M_BY_NODE                  = 00001000
MNR_PRO$L_PAGEFLTS              = 00000027              MRB$M_DISPLAY                  = 00000001
MNR_PRO$L_PCTINT                = 00000004              MRB$M_DISP_TO_FILE             = 00000020
MNR_PRO$L_PCTREC                = 00000000              MRB$M_DIS_CL_REQ               = 00000100
MNR_PRO$L_STS                   = 0000001F              MRB$M_INDEFEND                 = 00000010
MNR_PRO$L_UIC                   = 00000004              MRB$M_INP_CL_REQ               = 00000040
MNR_PRO$O_LNAME                 = 0000000B              MRB$M_MFSUM                    = 00000800
MNR_PRO$S_LNAME                 = 00000010              MRB$M_PLAYBACK                 = 00000008
MNR_PRO$S_PROCESS_CLASS         = 0000003B              MRB$M_PROC_REQ                 = 00004000
MNR_PRO$S_PRO_CLASS_PRE         = 00000008              MRB$M_RECORD                   = 00000002
MNR_PRO$W_GPGCNT                = 0000001B              MRB$M_REC_CL_REQ               = 00000080
MNR_PRO$W_PPGCNT                = 0000001D              MRB$M_SUMMARY                  = 00000004
MNR_PRO$W_STATE                 = 00000008              MRB$M_SUM_CL_REQ               = 00000200
MNR_SYI$B_MPCPUS                = 0000000D              MRB$M_SYSCLS                   = 00002000
MNR_SYI$B_TYPE                  = 00000000              MRB$O_CLASSBITS                = 00000032
MNR_SYI$K_BALSETMEM             = 0000001E              MRB$Q_BEGINNING                = 00000000
MNR_SYI$K_CPUTYPE               = 00000026              MRB$Q_ENDING                   = 00000008
MNR_SYI$K_MPWHILIM              = 00000022              MRB$S_BEGINNING                = 00000008
MNR_SYI$K_NODENAME              = 0000000E              MRB$S_CLASSBITS                = 00000010
MNR_SYI$K_SIZE                  = 0000002A              MRB$S_ENDING                   = 00000008
MNR_SYI$L_BALSETMEM             = 0000001E              MRB$S_FLAGS                    = 00000002
MNR_SYI$L_CPUTYPE               = 00000026              MRB$S_MRB                      = 00000045
MNR_SYI$L_MPWHILIM              = 00000022              MRB$V_ALL_CLASS                = 0000000A
MNR_SYI$Q_BOOTTIME              = 00000003              MRB$V_BY_NODE                  = 0000000C
MNR_SYI$S_BOOTTIME              = 00000008              MRB$V_DISPLAY                  = 00000000
MNR_SYI$S_FILLER                = 0000000E              MRB$V_DISP_TO_FILE             = 00000005
MNR_SYI$S_FLAGS                 = 00000002              MRB$V_DIS_CL_REQ               = 00000008
MNR_SYI$S_NODENAME              = 00000010              MRB$V_FILLER                   = 0000000F
MNR_SYI$S_SYS_INFO              = 0000002A              MRB$V_INDEFEND                 = 00000004
MNR_SYI$S_TYPE                  = 00000008              MRB$V_INP_CL_REQ               = 00000006
MNR_SYI$T_NODENAME              = 0000000E              MRB$V_MFSUM                    = 0000000B
MNR_SYI$V_CLUSMEM               = 00000000              MRB$V_PLAYBACK                 = 00000003
MNR_SYI$V_FILLER                = 00000002              MRB$V_PROC_REQ                 = 0000000E
MNR_SYI$V_RESERVED1             = 00000001              MRB$V_RECORD                   = 00000001
MNR_SYI$W_FLAGS                 = 00000001              MRB$V_REC_CL_REQ               = 00000007
MNR_SYI$W_MAXPRCCT              = 0000000B              MRB$V_SUMMARY                  = 00000002
MODES_CLSNO                       ********  X     03    MRB$V_SUM_CL_REQ               = 00000009
MODES_ICOUNT                      ********  X     03    MRB$V_SYSCLS                   = 0000000D
MON_ERR                           00001EB6 RG     03    MRB$W_CLASSCT                  = 00000030
MOVE_BARS                         00000AE7 R      03    MRB$W_FLAGS                    = 00000043
```

```
MRBPTR                           ********    X   03      QUAL$A_ALL                        = 00000064
MWAITLIST                        ********    X   03      QUAL$A_AVE                        = 00000074
NAMESTR                          000022C0 RG     01      QUAL$A_BEG                        = 00000004
NAME_COL                         00000000 RG     01      QUAL$A_BY_NODE                    = 00000054
NAME_COL_BAR                   = 00000002  G             QUAL$A_CLASS                      = 0000005C
NAME_COL_MFSUM                 = 00000001  G             QUAL$A_COMM                       = 0000004C
NAME_COL_TAB                   = 00000005  G             QUAL$A_CPU                        = 000000AC
NEWXPOS                          00000089 R      01      QUAL$A_CUR                        = 0000006C
NO                             = 00000000  G             QUAL$A_DISP                       = 00000034
NORMAL                           ********    X   03      QUAL$A_END                        = 0000000C
NUMB_ONLY                        ********    X   03      QUAL$A_FLUSH                      = 0000001C
OTHER_VID                      = 00000003                QUAL$A_INP                        = 0000002C
OUTDSC                           00001A03 RG     01      QUAL$A_INT                        = 00000014
OUTDSC_SIZE                    = 000006A4                QUAL$A_ITEM                       = 000000BC
PCB$V_RES                      = 00000000                QUAL$A_MAX                        = 00000084
PCENT_STR                        0000256C RG     01      QUAL$A_MIN                        = 0000007C
PCTEN                            00000E6F R      03      QUAL$A_PCENT                      = 000000B4
PC_BUFS                        = 00000004  G             QUAL$A_REC                        = 0000003C
PERFTABLE                        ********    X   03      QUAL$A_SUMM                       = 00000044
PLAY_STR                         000023C4 R      01      QUAL$A_TOPB                       = 0000009C
PREV_PD                          000000E4 R      01      QUAL$A_TOPC                       = 0000008C
PRINT_SCREEN                     0000168D R      03      QUAL$A_TOPD                       = 00000094
PROCDISPS                      = 00000005                QUAL$A_TOPF                       = 000000A4
PROCESS_CLASS                  = 00C00000                QUAL$A_VIEW                       = 00000024
PROCHEAD_STR                     00002403 RG     01      QUAL$L_ALL                        = 00000060
PROCS_CLSNO                      ********    X   03      QUAL$L_AVE                        = 00000070
PROCS_PER_REC                    00000016 R      01      QUAL$L_BEG                        = 00000000
PROC_LINE                      = 0000004B                QUAL$L_BY_NODE                    = 00000050
PROC_NRES_STR                    00002676 R      01      QUAL$L_CLASS                      = 00000058
PROC_RES_STR                     00002631 R      01      QUAL$L_COMM                       = 00000048
PROC_SETUP_STR                   00002616 R      01      QUAL$L_CPU                        = 000000A8
PROC_WRI_BUFD                    0000001A R      01      QUAL$L_CUR                        = 00000068
PROMPT_STR                       000000E8 RG     01      QUAL$L_DISP                       = 00000030
PRO_CLASS_PRE                  = 00000000                QUAL$L_END                        = 00000008
PTS_ESCB                         00001BC5 R      03      QUAL$L_FLUSH                      = 00000018
PTS_ESCF                         00001C28 R      03      QUAL$L_INP                        = 00000028
PTS_ESCG                         00001C35 R      03      QUAL$L_INT                        = 00000010
PTS_ESCH                         00001C1B R      03      QUAL$L_ITEM                       = 000000B8
PTS_ESCJ                         00001C12 R      03      QUAL$L_MAX                        = 00000080
PTS_ESCK                         00001C09 R      03      QUAL$L_MIN                        = 00000078
PTS_ESCL                         00001BCE R      03      QUAL$L_PCENT                      = 000000B0
PTS_ESCR                         00001BDE R      03      QUAL$L_REC                        = 00000038
PTS_ESCU                         00001BE7 R      03      QUAL$L_SUMM                       = 00000040
PTS_ESCY                         00001BEE R      03      QUAL$L_TOPB                       = 00000098
PTS_RET                          00001C73 R      03      QUAL$L_TOPC                       = 00000088
PTS_STAT                         0000008D R      01      QUAL$L_TOPD                       = 00000090
PUTMSGSIZE                     = 0000001A                QUAL$L_TOPF                       = 000000A0
PUTMSGVEC                        00002741 R      01      QUAL$L_VIEW                       = 00000020
PUTSCRARG                        0000271B R      01      QUAL$S_QUALIFIER_DESC             = 000000C0
PUTS_ALTSET                    = 00000000                QUALIFIER_DESC                    = 00000000
PUTS_REGSET                    = 00000000                READ_INPUT                          ********    X   03
PUT_ESC_SEQ                      00001C7B R      03      REC_STR                             000023E5 R      01
PUT_TO_SCREEN                    00001B5D RG     03      REG                                 000012C1 R      03
QEZ_RET                          00000469 R      03      REGSET                              ********    X   03
QLQ_RET                          00000452 R      03      REG_BUFS                          = 00000004  G
QUAD_EQ_0                        00000453 RG     03      REG_PROC                          = 00000000
QUAD_LT_QUAD                     00000436 RG     03      REG_SET                             00002711 R      01
```

```
RPB$L_BOOTR5                  = 00000030              SS$_BADATTRIB                 = 00000034   G
RPB$V_MPM                     = 0000000B              SS$_BADCHKSUM                 = 00000808   G
RSN$_MAX                      = 0000000F              SS$_BADESCAPE                 = 0000003C   G
RWAITLIST                       ********    X  03     SS$_BADFILEHDR                = 00000810   G
SCANBUF                         00001B63 R     03     SS$_BADFILENAME               = 00000818   G
SCB$B_FLAGS                   = 00000002              SS$_BADFILEVER                = 00000820   G
SCB$K_SIZE                    = 00000003              SS$_BADIMGHDR                 = 00000044   G
SCB$S_FILLER                  = 00000006              SS$_BADIRECTORY               = 00000828   G
SCB$S_FLAGS                   = 00000001              SS$_BADISD                    = 00002004   G
SCB$S_STATS_BLOCK             = 00000003              SS$_BADPARAM                  = 00000014   G
SCB$V_ACTIVE                  = 00000001              SS$_BADQFILE                  = 000003BC   G
SCB$V_CURRENT                 = 00000000              SS$_BADQUEUEHDR               = 00000394   G
SCB$V_FILLER                  = 00000002              SS$_BADRCT                    = 0000216C   G
SCB$W_DBIDX                   = 00000000              SS$_BADSTACK                  = 000002B4   G
SCH$C_MWAIT                   = 00000002              SS$_BADVEC                    = 00002064   G
SCR$B_DEVTYPE                 = 00000008              SS$_BEGOFFILE                 = 00000938   G
SCR$ERASE_LINE                  ********    X  03     SS$_BEGOFTAPE                 = 00000A10   G
SCR$ERASE_PAGE                  ********    X  03     SS$_BLOCKCNTERR               = 00000940   G
SCR$L_FLAGS                   = 00000000              SS$_BREAK                     = 00000414   G
SCR$PUT_LINE                    ********    X  03     SS$_BUFBYTALI                 = 0000030C   G
SCR$PUT_SCREEN                  ********    X  03     SS$_BUFFEROVF                 = 00000601   G
SCR$SCREEN_INFO                 ********    X  03     SS$_BUFNOTALIGN               = 00000324   G
SCR$SET_CURSOR                  ********    X  03     SS$_BUGCHECK                  = 000002A4   G
SCR$SET_OUTPUT                  ********    X  03     SS$_CANCEL                    = 00000830   G
SCR$V_BOLD                    = 00000000              SS$_CANCELGRANT               = 00000E2A   G
SCR$V_DECCRT                  = 00000006              SS$_CHAINW                    = 00000C0B   G
SCR$V_REVERSE                 = 00000001              SS$_CHANINTLK                 = 0000004C   G
SCR$V_SCREEN                  = 00000000              SS$_CLEARED                   = 00002104   G
SCR$V_UNDERLINE               = 00000003              SS$_CLIFRCEXT                 = 00000980   G
SCRDSC                          000020AF RG    01     SS$_CMODSUPR                  = 0000041C   G
SCRDSC_SIZE                   = 00000200              SS$_CMODUSER                  = 00000424   G
SCS_DISPNAM                     00002166 RG    03     SS$_COMMHARD                  = 000020C4   G
SCS_FAO                         000025E6 RG    01     SS$_COMPAT                    = 0000042C   G
SELECT_REV                      00001CDC R     03     SS$_CONCEALED                 = 00000691   G
SELECT_REV_LEVS                 00001C98 RG    03     SS$_CONNECFAIL                = 000020DC   G
SELECT_SET                      00001C40 R     03     SS$_CONTINUE                  = 00000001   G
SHR$_OPENIN                   = 00001098              SS$_CONTROLC                  = 00000651   G
SI                            = 0000000F              SS$_CONTROLO                  = 00000609   G
SIGNALED_ERR                    00001F0D RG    03     SS$_CONTROLY                  = 00000611   G
SIGNAL_MON_ERR                  00001F54 RG    03     SS$_CREATED                   = 00000619   G
SKIP_TO_CLASS                   ********    X  03     SS$_CTRLERR                   = 00000054   G
SORT_PROCS                      00001346 R     03     SS$_CVTUNGRANT                = 0000213C   G
SPTR                            ********    X  03     SS$_DATACHECK                 = 0000005C   G
SR_RSB                          00001D48 R     03     SS$_DATALATE                  = 00002274   G
SS$_ABORT                     = 0000002C  G           SS$_DATAOVERUN                = 00000838   G
SS$_ACCONFLICT                = 00000800  G           SS$_DBGEVENT                  = 000006C1   G
SS$_ACCVIO                    = 0000000C  G           SS$_DBGOPCREQ                 = 000006A1   G
SS$_ACEEXISTS                 = 00000E42  G           SS$_DEADLOCK                  = 00000E0A   G
SS$_ACEIDMATCH                = 000006B9  G           SS$_DEBUG                     = 0000046C   G
SS$_ACLEMPTY                  = 000009D0  G           SS$_DECOVF                    = 000004A4   G
SS$_ACLFULL                   = 000009F8  G           SS$_DEVACTIVE                 = 000002C4   G
SS$_ACPVAFUL                  = 000002FC  G           SS$_DEVALLOC                  = 00000840   G
SS$_ALRDYCLOSED               = 000006A9  G           SS$_DEVALRALLOC               = 00000641   G
SS$_ARTRES                    = 00000474  G           SS$_DEVASSIGN                 = 00000848   G
SS$_ASTFLT                    = 0000040C  G           SS$_DEVCMDERR                 = 0000032C   G
SS$_BADACL                    = 00000E3A  G           SS$_DEVFOREIGN                = 00000064   G
SS$_BADACLCTX                 = 000021CC  G           SS$_DEVICEFULL                = 00000850   G
```

```
SS$_DEVINACT                    = 000020D4  G             SS$_FILNOTEXP                   = 000000B4  G
SS$_DEVMOUNT                    = 0000006C  G             SS$_FLTDIV                      = 00000494  G
SS$_DEVNOTALLOC                 = 00000858  G             SS$_FLTDIV_F                    = 000004BC  G
SS$_DEVNOTDISM                  = 000021B4  G             SS$_FLTOVF                      = 0000048C  G
SS$_DEVNOTMBX                   = 00000074  G             SS$_FLTOVF_F                    = 000004B4  G
SS$_DEVNOTMOUNT                 = 0000007C  G             SS$_FLTUND                      = 0000049C  G
SS$_DEVOFFLINE                  = 00000084  G             SS$_FLTUND_F                    = 000004C4  G
SS$_DEVREQERR                   = 00000334  G             SS$_FORCEDERROR                 = 00002144  G
SS$_DGQINCOMP                   = 000009C0  G             SS$_FORCEDEXIT                  = 0000217C  G
SS$_DIRALLOC                    = 000009C8  G             SS$_FORMAT                      = 000000BC  G
SS$_DIRFULL                     = 00000860  G             SS$_GPTFULL                     = 000000C4  G
SS$_DIRNOTEMPTY                 = 00002174  G             SS$_GSDFULL                     = 000000CC  G
SS$_DISCONNECT                  = 0000204C  G             SS$_HANGUP                      = 000002CC  G
SS$_DRVERR                      = 0000008C  G             SS$_HEADERFULL                  = 000008C8  G
SS$_DUPDSKQUOTA                 = 000003DC  G             SS$_IDMISMATCH                  = 000003F4  G
SS$_DUPFILENAME                 = 00000868  G             SS$_IDXFILEFULL                 = 000008D0  G
SS$_DUPIDENT                    = 0000222C  G             SS$_ILLBLKNUM                   = 000000DC  G
SS$_DUPLNAM                     = 00000094  G             SS$_ILLCDTST                    = 00002154  G
SS$_DUPUNIT                     = 000021C4  G             SS$_ILLCNTRFUNC                 = 000000E4  G
SS$_ENDOFFILE                   = 00000870  G             SS$_ILLEFC                      = 000000EC  G
SS$_ENDOFTAPE                   = 00000878  G             SS$_ILLIOFUNC                   = 000000F4  G
SS$_ENDOFUSRLBL                 = 00000970  G             SS$_ILLLBLAST                   = 00000968  G
SS$_ENDOFVOLUME                 = 000009A0  G             SS$_ILLPAGCNT                   = 000000FC  G
SS$_EOTIN                       = 00000C03  G             SS$_ILLSELF                     = 0000214C  G
SS$_EXASTLM                     = 00002A04  G             SS$_ILLSEQOP                    = 000002DC  G
SS$_EXBIOLM                     = 00002A0C  G             SS$_ILLSER                      = 00000104  G
SS$_EXBYTLM                     = 00002A14  G             SS$_ILLUSRLBLRD                 = 00000958  G
SS$_EXCPUTIM                    = 000020AC  G             SS$_ILLUSRLBLWT                 = 00000960  G
SS$_EXDEPTH                     = 00000E1A  G             SS$_INCOMPAT                    = 00000699  G
SS$_EXDIOLM                     = 00002A1C  G             SS$_INCSEGTRA                   = 00002234  G
SS$_EXDISKQUOTA                 = 000003EC  G             SS$_INCSHAMEM                   = 000022CC  G
SS$_EXENQLM                     = 00002A44  G             SS$_INCVOLLABEL                 = 0000010C  G
SS$_EXFILLM                     = 00002A24  G             SS$_INHCHME                     = 000004D4  G
SS$_EXGBLPAGFIL                 = 00002164  G             SS$_INHCHMK                     = 000004CC  G
SS$_EXLNMQUOTA                  = 0000224C  G             SS$_INSFARG                     = 00000114  G
SS$_EXPGFLQUOTA                 = 00002A2C  G             SS$_INSFBUFDP                   = 0000033C  G
SS$_EXPORTQUOTA                 = 000003AC  G             SS$_INSFCDT                     = 000021AC  G
SS$_EXPRCLM                     = 00002A34  G             SS$_INSFMAPREG                  = 00000344  G
SS$_EXQUOTA                     = 0000001C  G             SS$_INSFMEM                     = 00000124  G
SS$_EXQUOTAEND                  = 00002AFF  G             SS$_INSFRAME                    = 0000012C  G
SS$_EXQUOTASTRT                 = 00002A00  G             SS$_INSFSPTS                    = 00002044  G
SS$_EXTIDXFILE                  = 00000880  G             SS$_INSFWSL                     = 0000011C  G
SS$_EXTQELM                     = 00002A3C  G             SS$_INSSWAPSPACE                = 00002264  G
SS$_FCPREADERR                  = 00000888  G             SS$_INTDIV                      = 00000484  G
SS$_FCPREPSTN                   = 00000988  G             SS$_INTERLOCK                   = 0000038C  G
SS$_FCPREWNDERR                 = 00000890  G             SS$_INTOVF                      = 0000047C  G
SS$_FCPSPACERR                  = 00000898  G             SS$_INVEXHLIST                  = 000022B4  G
SS$_FCPWRITERR                  = 000008A0  G             SS$_INVLOGIN                    = 0000209C  G
SS$_FILACCERR                   = 0000009C  G             SS$_INVSECLASS                  = 000022C4  G
SS$_FILALRACC                   = 000000A4  G             SS$_IVACL                       = 000021E4  G
SS$_FILELOCKED                  = 000008A8  G             SS$_IVADDR                      = 00000134  G
SS$_FILENUMCHK                  = 000008B0  G             SS$_IVBUFLEN                    = 0000034C  G
SS$_FILEPURGED                  = 00000679  G             SS$_IVCHAN                      = 0000013C  G
SS$_FILESEQCHK                  = 000008B8  G             SS$_IVCHAR                      = 000020CC  G
SS$_FILESTRUCT                  = 000008C0  G             SS$_IVCHNLSEC                   = 0000026C  G
SS$_FILNOTACC                   = 000000AC  G             SS$_IVDEVNAM                    = 00000144  G
SS$_FILNOTCNTG                  = 000002AC  G             SS$_IVGSDNAM                    = 0000014C  G
```

```
SS$_IVIDENT                     = 00002224  G          SS$_NOLCLMEDA                   = 000021F4  G
SS$_IVLOCKID                    = 00002124  G          SS$_NOLICENSE                   = 00002´94  G
SS$_IVLOGNAM                    = 00000154  G          SS$_NOLINKS                     = 0000027C  G
SS$_IVLOGTAB                    = 0000015C  G          SS$_NOLISTENER                  = 0000215C  G
SS$_IVLVEC                      = 0000203C  G          SS$_NOLOCKID                    = 00000E12  G
SS$_IVMODE                      = 00000354  G          SS$_NOLOGNAM                    = 000001BC  G
SS$_IVPROTECT                   = 000002F4  G          SS$_NOLOGTAB                    = 00002294  G
SS$_IVQUOTAL                    = 00000164  G          SS$_NOLOG_IO                    = 0000283C  G
SS$_IVSECFLG                    = 0000016C  G          SS$_NOMBX                       = 00000274  G
SS$_IVSECIDCTL                  = 000002E4  G          SS$_NOMOREACE                   = 000009E0  G
SS$_IVSSRQ                      = 00000174  G          SS$_NOMOREFILES                 = 00000930  G
SS$_IVSTSFLG                    = 0000017C  G          SS$_NOMORELOCK                  = 00000A08  G
SS$_IVTIME                      = 00000184  G          SS$_NOMORENODE                  = 00000A00  G
SS$_JBCERROR                    = 0000218C  G          SS$_NOMOREPROC                  = 000009A8  G
SS$_KERNELINV                   = 00002244  G          SS$_NOMOUNT                     = 0000288C  G
SS$_LCKPAGFUL                   = 000000D4  G          SS$_NONETMBX                    = 000028A4  G
SS$_LENVIO                      = 0000018C  G          SS$_NONEXDRV                    = 000001C4  G
SS$_LINEABRT                    = 00000E02  G          SS$_NONEXPR                     = 000008E8  G
SS$_LINKABORT                   = 000020E4  G          SS$_NONLOCAL                    = 000008F0  G
SS$_LINKDISCON                  = 000020EC  G          SS$_NOOPER                      = 00002894  G
SS$_LINKEXIT                    = 000020F4  G          SS$_NOP1VA                      = 00002024  G
SS$_LKWSETFUL                   = 00000194  G          SS$_NOPFNMAP                    = 000028D4  G
SS$_LNMCREATED                  = 000006B1  G          SS$_NOPHY_IO                    = 000028B4  G
SS$_MBFULL                      = 000008D8  G          SS$_NOPRIV                      = 00000024  G
SS$_MBTOOSML                    = 0000019C  G          SS$_NOPRIVEND                   = 000029FF  G
SS$_MCHECK                      = 000002BC  G          SS$_NOPRIVSTRT                  = 00002800  G
SS$_MCNOTVALID                  = 0000035C  G          SS$_NOPRMCEB                    = 00002854  G
SS$_MEDOFL                      = 000001A4  G          SS$_NOPRMGBL                    = 000028C4  G
SS$_MSGNOTFND                   = 00000621  G          SS$_NOPRMJNL                    = 00002904  G
SS$_MTLBLLONG                   = 00000304  G          SS$_NOPRMMBX                    = 0000285C  G
SS$_MULTRMS                     = 0000202C  G          SS$_NOPSWAPM                    = 00002864  G
SS$_MUSTCLOSEFL                 = 00000948  G          SS$_NOQFILE                     = 000003C4  G
SS$_NOACLSUPPORT                = 000022BC  G          SS$_NOREADALL                   = 00002924  G
SS$_NOACNT                      = 0000284C  G          SS$_NOREGAVIL                   = 000021FC  G
SS$_NOALLSPOOL                  = 00002824  G          SS$_NOREGSUIT                   = 00002204  G
SS$_NOALTPRI                    = 0000286C  G          SS$_NORMAL                      = 00000001  G
SS$_NOAQB                       = 00000314  G          SS$_NOSECURITY                  = 00002934  G
SS$_NOBUGCHK                    = 000028BC  G          SS$_NOSETPRV                    = 00002874  G
SS$_NOBYPASS                    = 000028EC  G          SS$_NOSHARE                     = 0000292C  G
SS$_NOCMEXEC                    = 0000280C  G          SS$_NOSHMBLOCK                  = 000003B4  G
SS$_NOCMKRNL                    = 00002804  G          SS$_NOSHMEM                     = 000028DC  G
SS$_NODATA                      = 000001AC  G          SS$_NOSHRIMG                    = 000021BC  G
SS$_NODELEAVE                   = 0000223C  G          SS$_NOSIGNAL                    = 00000900  G
SS$_NODETACH                    = 0000282C  G          SS$_NOSLOT                      = 0000039C  G
SS$_NODEVAVL                    = 000009B0  G          SS$_NOSOLICIT                   = 00000284  G
SS$_NODIAGNOSE                  = 00002834  G          SS$_NOSUCHDEV                   = 00000908  G
SS$_NODISKQUOTA                 = 000003E4  G          SS$_NOSUCHFILE                  = 00000910  G
SS$_NODOWNGRADE                 = 0000290C  G          SS$_NOSUCHID                    = 000021EC  G
SS$_NOENTRY                     = 000009D8  G          SS$_NOSUCHNODE                  = 0000028C  G
SS$_NOEXQUOTA                   = 0000289C  G          SS$_NOSUCHOBJ                   = 000020A4  G
SS$_NOFILACC                    = 000022AC  G          SS$_NOSUCHPGM                   = 0000220C  G
SS$_NOGROUP                     = 00002844  G          SS$_NOSUCHSEC                   = 00000978  G
SS$_NOGRPNAM                    = 0000281C  G          SS$_NOSUCHUSER                  = 00002084  G
SS$_NOGRPPRV                    = 0000291C  G          SS$_NOSYSGBL                    = 000028CC  G
SS$_NOHANDLER                   = 000008F8  G          SS$_NOSYSLCK                    = 000028F4  G
SS$_NOHOMEBLK                   = 000008E0  G          SS$_NOSYSNAM                    = 00002814  G
SS$_NOIOCHAN                    = 000001B4  G          SS$_NOSYSPRV                    = 000028E4  G
```

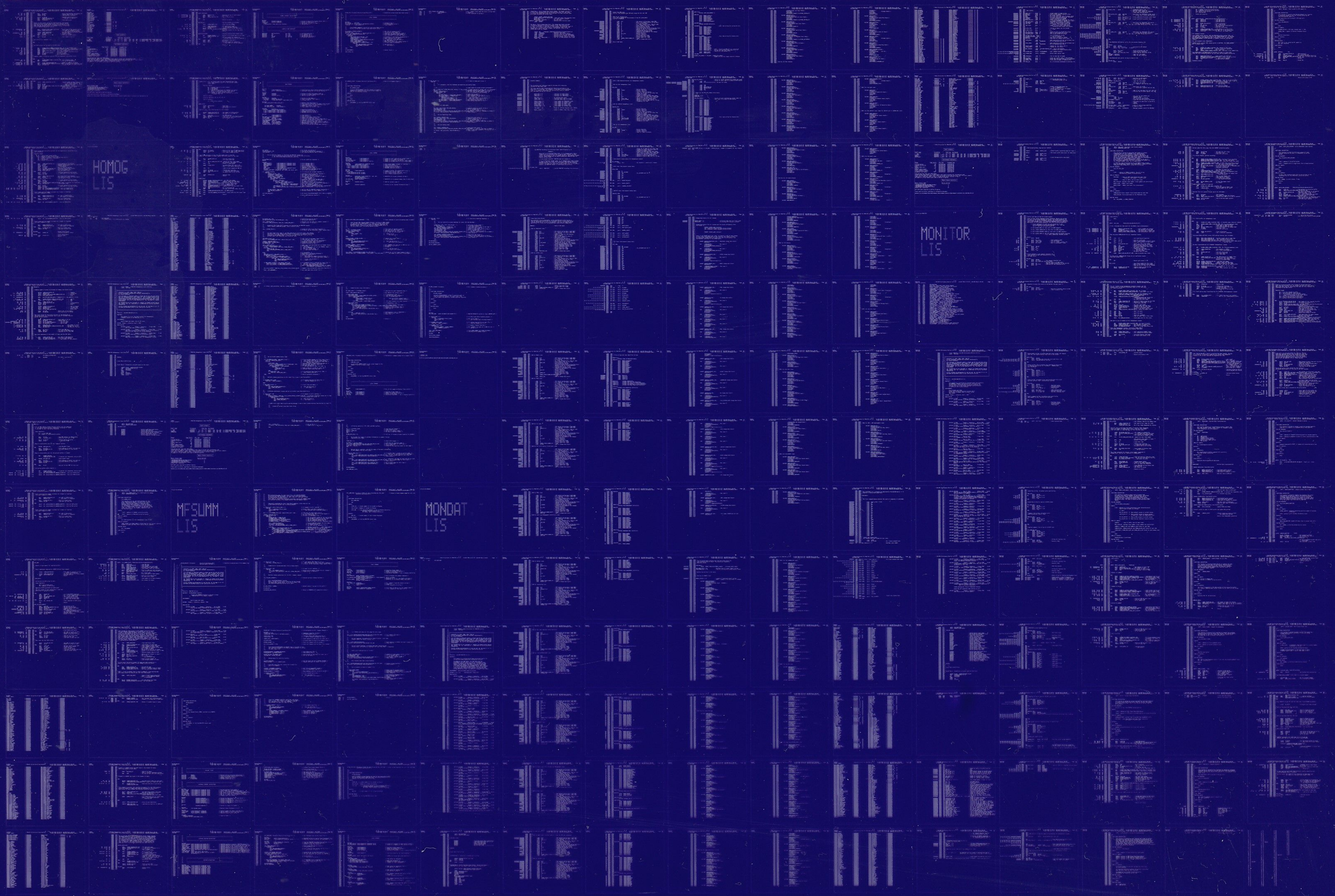
```
SS$_NOALLPRIV                    = 00000681  G              SS$_RIGHTSFULL                  = 000009E8  G
SS$_NOTAPEOP                     = 00000264  G              SS$_ROPRAND                     = 00000454  G
SS$_NOTCREATOR                   = 00000384  G              SS$_SECTBLFUL                   = 0000021C  G
SS$_NOTFILEDEV                   = 000001CC  G              SS$_SERIOUSEXCP                 = 000021D4  G
SS$_NOTINSTALL                   = 00002014  G              SS$_SHACHASTA                   = 00002284  G
SS$_NOTINTBLSZ                   = 000001D4  G              SS$_SHACPYINP                   = 000022D4  G
SS$_NOTLABELMT                   = 000001DC  G              SS$_SHARTOOBIG                  = 0000201C  G
SS$_NOTMODIFIED                  = 00000659  G              SS$_SHMGSNOTMAP                 = 0000036C  G
SS$_NOTMPJNL                     = 000028FC  G              SS$_SHMNOTCNCT                  = 0000037C  G
SS$_NOTMPMBX                     = 0000287C  G              SS$_SHRIDMISMAT                 = 000020BC  G
SS$_NOTNETDEV                    = 000002EC  G              SS$_SHUT                        = 0000208C  G
SS$_NOTPRINTED                   = 00002184  G              SS$_SSFAIL                      = 0000045C  G
SS$_NOTQUEUED                    = 000009B8  G              SS$_SUBLOCKS                    = 0000212C  G
SS$_NOTRAN                       = 00000629  G              SS$_SUBRNG                      = 000004AC  G
SS$_NOTSQDEV                     = 000001E4  G              SS$_SUPERSEDE                   = 00000631  G
SS$_NOTVOLSET                    = 00000998  G              SS$_SUSPENDED                   = 000003A4  G
SS$_NOUPGRADE                    = 00002914  G              SS$_SYNCH                       = 00000689  G
SS$_NOVOLACC                     = 000022A4  G              SS$_SYSAPMAX                    = 00007FFF  G
SS$_NOVOLPRO                     = 000028AC  G              SS$_SYSAPMIN                    = 00007E00  G
SS$_NOWORLD                      = 00002884  G              SS$_SYSVERDIF                   = 00000671  G
SS$_NOWRT                        = 000003FC  G              SS$_TAPEPOSLOST                 = 00000224  G
SS$_OPCCUS                       = 00000434  G              SS$_TBIT                        = 00000464  G
SS$_OPCDEC                       = 0000043C  G              SS$_TEMPLATEDEV                 = 000021DC  G
SS$_OPINCOMPL                    = 000002D4  G              SS$_TERMNETDEV                  = 0000228C  G
SS$_OPRABORT                     = 000020B4  G              SS$_THIRDPARTY                  = 0000207C  G
SS$_OVRDSKQUOTA                  = 00000669  G              SS$_TIMEOUT                     = 0000022C  G
SS$_OVRMAXARG                    = 0000227C  G              SS$_TOOMANYLNAM                 = 00000374  G
SS$_PAGOWNVIO                    = 000001EC  G              SS$_TOOMANYREDS                 = 0000211C  G
SS$_PAGRDERR                     = 00000444  G              SS$_TOOMANYVER                  = 00000990  G
SS$_PARENT_DEL                   = 00002254  G              SS$_TOOMUCHDATA                 = 0000029C  G
SS$_PARITY                       = 000001F4  G              SS$_UNASEFC                     = 00000234  G
SS$_PARNOTGRANT                  = 00002134  G              SS$_UNREACHABLE                 = 00002094  G
SS$_PARNOTSYS                    = 0000225C  G              SS$_UNSAFE                      = 0000023C  G
SS$_PARTESCAPE                   = 000001FC  G              SS$_UNSOLICIT                   = 00002114  G
SS$_PARTMAPPED                   = 00000E22  G              SS$_UNWIND                      = 00000920  G
SS$_PATHLOST                     = 000020FC  G              SS$_UNWINDING                   = 00000928  G
SS$_PFMBSY                       = 00000204  G              SS$_VALNOTVALID                 = 000009F0  G
SS$_PGMLDFAIL                    = 00002214  G              SS$_VASFULL                     = 00000244  G
SS$_PGMSTDALN                    = 0000221C  G              SS$_VCBROKEN                    = 0000219C  G
SS$_PLHLDR                       = 00000404  G              SS$_VCCLOSED                    = 000021A4  G
SS$_POWERFAIL                    = 00000364  G              SS$_VECFULL                     = 00002034  G
SS$_PRIVINSTALL                  = 00002054  G              SS$_VECINUSE                    = 0000024C  G
SS$_PROTINSTALL                  = 0000205C  G              SS$_VOLINV                      = 00000254  G
SS$_PROTOCOL                     = 00002074  G              SS$_VOLOERR                     = 0000226C  G
SS$_PSTFULL                      = 0000020C  G              SS$_WAITUSRLBL                  = 00000950  G
SS$_QFACTIVE                     = 000003CC  G              SS$_WASCLR                      = 00000001  G
SS$_QFNOTACT                     = 000003D4  G              SS$_WASECC                      = 00000639  G
SS$_RADRMOD                      = 0000044C  G              SS$_WASSET                      = 00000009  G
SS$_RDDELDATA                    = 00000661  G              SS$_WRITLCK                     = 0000025C  G
SS$_REJECT                       = 00000294  G              SS$_WRONGACP                    = 0000031C  G
SS$_RELINK                       = 0000200C  G              SS$_WRONGNAME                   = 0000229C  G
SS$_REMOTE                       = 00000649  G              STACK_TOP                         000011C1 R        03
SS$_REMRSRC                      = 0000206C  G              STARTPOS                        = 00000032
SS$_RESET                        = 0000210C  G              START_XPOS                      = 00000026
SS$_RESIGNAL                     = 00000918  G              STATELIST                         ********    X     03
SS$_RESULTOVF                    = 00000214  G              STATES_CLSNO                      ********    X     03
SS$_RETRY                        = 00000E32  G              STATHEAD_STR                      0000251E RG       01
```

```
STATS                          = 00000005
STATS_BLOCK                    = 00000000
STATUS_PARMS                     000023F7 RG      01
STATUS_STR                       000022FF RG      01
STAT_HEAD                        00002570 RG      01
STAT_LONG                        0000257C RG      01
STS$K_ERROR                    = 00000002
STS$S_FAC_NO                   = 0000000C
STS$V_FAC_NO                   = 00000010
SUMMARY_INIT                     00000A5B RG      03
SUMMARY_TOP                      0000145D RG      03
SUMMLINE_STR                     00002371 RG      01
SUMM_STR                         000023D3 R       01
SYI$_CPU                       = 00002000
SYI$_NODENAME                  = 000010D9
SYS$ASCTIM                       ********  GX     03
SYS$ASSIGN                       ********  GX     03
SYS$CMKRNL                       ********  GX     03
SYS$FAOL                         ********  GX     03
SYS$GETCHN                       ********  GX     03
SYS$GETSYIW                      ********  GX     03
SYS$GETTIM                       ********  GX     03
SYS$PUTMSG                       ********  GX     03
SYS$QIOW                         ********  GX     03
SYS$SETIMR                       ********  GX     03
SYS$TRNLOG                       ********  GX     03
SYS$WAITFR                       ********  GX     03
SYSCMD_DESC                      000000B1 R       01
SYSNOD_NAM                       000000C4 R       01
SYSOUT_TYPE                      00002615 R       01
SYSTEM$_FACILITY               = 00000000  G
SYSTEM_CLSNO                     ********   X     03
SYS_BOX_STR_ADDR                 00000085 RG      01
SYS_BOX_STR_G                    ********   X     03
SYS_BOX_STR_H                    ********   X     03
SYS_BOX_STR_LEN                  00000083 RG      01
SYS_BOX_STR_LEN_G                ********   X     03
SYS_BOX_STR_LEN_H                ********   X     03
SYS_DATA_ADDR                    0000007B RG      01
SYS_DATA_LEN                     0000007F RG      01
SYS_FAC_NO                     = 00000000
SYS_FAO_STR                      ********   X     03
SYS_HEAD_STR                     000024BD RG      01
SYS_INFO                       = 00000000
SYS_SUMMLINE_STR                 0000239E RG      01
SYS_TIME_STR                     00002362 RG      01
SYS_TOP_VEC                      0000003B RG      01
S_TOP_TICKS                      00000037 R       01
S_TOP_TIME                       0000002F R       01
TABHEAD_STR                      00002479 RG      01
TAB_LWORDS                     = 00000008
TEMP_4_BLOCK                   = 00000000
TIME_STR                         00002353 RG      01
TITLE_STR                        00002315 RG      01
TM4$A_BUFFERS                  = 00000008
TM4$A_ITMSTR                   = 00000004
TM4$K_SIZE                     = 00000010

TM4$L_ECOUNT                   = 00000000
TM4$L_FLTSECS                  = 0000000C
TM4$S_TEMP_4_BLOCK             = 00000010
TOPB                             000012CB R       03
TOPBAR                         = 000026EA R       01
TOPB_PROC                      = 00000003
TOPC                             000012C1 R       03
TOPC_PROC                      = 00000001
TOPD                             000012C6 R       03
TOPD_PROC                      = 00000002
TOPF                             000012D0 R       03
TOPF_PROC                      = 00000004
TOPLNNO                        = 000026C6 R       01
TOPSTR                           000026C3 R       01
TOP_DIFFS                        000013ED R       03
TOP_PROCS                        00000022 R       01
TOP_TICKS                        0000002B R       01
TOP_TIME                         00000023 R       01
TRANSFORMS                       000003F0 R       03
TXT_DESC                         0000272F R       01
TXT_LENGTH                       0000272F R       01
TXT_START                        00002733 R       01
UNKNOWN_NODE                     000025F1 R       01
UPD_PC_MIN_MAX                   00000971 R       03
VID_BAR                        = 00000061
VIEWING_DEL                      ********   X     03
VT100_ALTSET                     00002702 R       01
VT100_CURSET                     0000270D R       01
VT100_REGSET                     000026F6 R       01
VT55CWIDTH                     = 0000004A  G
VT55HEIGHT                     = 000000EC
VT55WIDTH                      = 00000200
VT55XINCR                        00C000D4 RG      01
VT5X                           = 00000001
VTDATALINES                    = 0000000F  G
VTHEIGHT                       = 00000018  G
VTWIDTH                        = 00000050  G
WPR_RET                          00000586 R       03
WRITE_HEADER                     ********   X     03
WRITE_PROC_RECORDS               0000048A RG      03
WRITE_RECORD                     ********   X     03
YES                            = 00000001  G
```

```
                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+

PSECT name                Allocation          PSECT No.  Attributes
----------                ----------          ---------  ----------
.  ABS  .                 00000000  (      0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
MONDATA                   000027A9  (  10153.)  01 (  1.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD     WRT NOVEC QUAD
$ABS$                     00000000  (      0.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$MONCODE                 00002175  (   8565.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                      +------------------------+
                                      ! Performance indicators !
                                      +------------------------+

Phase                     Page faults   CPU Time       Elapsed Time
-----                     -----------   --------       ------------
Initialization                    48    00:00:00.13    00:00:01.28
Command processing               161    00:00:00.94    00:00:05.54
Pass 1                          1015    00:00:41.65    00:01:30.83
Symbol table sort                  0    00:00:05.61    00:00:08.40
Pass 2                           446    00:00:16.25    00:00:40.35
Symbol table output                2    00:00:00.96    00:00:01.81
Psect synopsis output              0    00:00:00.03    00:00:00.03
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals            1675    00:01:05.57    00:02:28.29

The working set limit was 3300 pages.
252219 bytes (493 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3159 non-local and 394 local symbols.
7063 source lines were read in Pass 1, producing 157 object records in Pass 2.
67 pages of virtual memory were used to define 55 macros.

                                     +--------------------------+
                                     ! Macro library statistics !
                                     +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[MONTOR.OBJ]MONLIB.MLB;1                11
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       8
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   32
TOTALS (all libraries)                              51

3099 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MONITOR/OBJ=OBJ$:MONITOR MSRC$:MONITOR/UPDATE=(ENH$:MONITOR)+EXECML$/LIB+LIB$:MONLIB/LIB
```

0240 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0241 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY